Full-domain Arithmetic ABE via Lookup Table Evaluation over BGG+ Encodings

Abstract.

1 Introduction

2 Technical Overview

We provide an overview of the techniques introduced in this paper, assuming familiarity with standard linear algebra and lattice notation. This discussing keeps things deliberately informal. The reader should refer to the technical sections for more precise definitions.

2.1 Review of BGG+ encodings and homomorphic operations.

We first provide a review of the BGG+ key-homomorphic encoding and its native operations. We use the BGG+ key-homomorphic encoding in the ring $\mathcal{R}_q$. For example, the encoding for the polynomial $z(X) \in \mathcal{R}_q$ is equivalent to

$$\mathbf{s}^T(\mathbf{a}^T - \mathbf{g}^T z(X)) + \mathbf{e}^T$$

The following algorithms are available to perform homomorphic addition, scalar multiplication and multiplication on BGG+ encodings:

- $\mathsf{EvalAdd}(\mathbf{a}_u, \mathbf{a}_v)$, $\mathsf{EvalScalar}(\alpha, \mathbf{a}_u)$ and $\mathsf{EvalMul}(\mathbf{a}_u, \mathbf{a}_v)$
- $\mathsf{EvalAddX}(\mathbf{a}_u, \mathbf{a}_v, z_u(X), z_v(X))$, $\mathsf{EvalScalarX}(\alpha, \mathbf{a}_u, z_u(X))$ and $\mathsf{EvalMulX}(\mathbf{a}_u, \mathbf{a}_v, z_u(X), z_v(X))$

Such that, considering the multiplication, the following key-homomorphic relation holds:

$$\begin{aligned}&\left(\mathbf{s}^T(\mathbf{a}_u^T - z_u(X)\mathbf{g}^T) + \mathbf{e}_u^T, \mathbf{s}^T(\mathbf{a}_v^T - z_v(X)\mathbf{g}^T) + \mathbf{e}_v^T\right) \mathbf{H}_{\times,w,z} \\ =&\mathbf{s}^T(\mathbf{a}_w^T - z_w(X)\mathbf{g}^T) + \mathbf{e}_w^T,\end{aligned}$$

Where $\mathbf{a}_w$ is the output of $\mathsf{EvalMul}$, $\mathbf{H}_{\times,w,z}$ is the output of $\mathsf{EvalMulX}$ and the error $\mathbf{e}_w$ of the resulting encoding grows linearly with the infinity norm of the matrix $\mathbf{H}_{\times,w,z}$. Similar relations hold for addition and scalar multiplication. Notably, the infinity norm of the matrix $\mathbf{H}_{\times,w,z}$ depends on the infinity norm of left hand side operand $z_u(X)$, such that the left operand must be bounded by an integer p for some $p < q$ to limit the growth of the error term after the multiplication. Because of this limitation, BGG+ encodings do not support evaluation of arithmetic circuits over $\mathcal{R}_q$ but, in practice, only supports binary circuits (i.e., circuits where multiplication gates have inputs in $\{0, 1\}$). In the next section we introduce a novel homomorphic gate for evaluating lookup tables (LUT) that will allow us to bypass this limitation.

2.2 Lookup evaluation gate over BGG+.

We introduce a lookup evaluation gate for a public lookup table $\mathcal{L} = \{(\bar{x}_k(X), \bar{y}_k(X))\}_{k\in[R]}$. Intuitively, this allows an evaluator to substitute their encoding of $z_u(X)$ with an encoding of $z_w(X) := \mathsf{LUT}_{\mathcal{L}}^{\mathsf{out}}(z_u(X))$, where $\mathsf{LUT}_{\mathcal{L}}^{\mathsf{out}}(z(X))$ returns $\bar{y}_k(X)$ if $z(X) = \bar{x}_k(X)$ for some $k \in [R]$ and $\perp$ otherwise.

More specifically, given a BGG+ encoding with public key $\mathbf{a}_u \in R_q^{m_g}$ for polynomial $z_u(X) \in R_q$, a public vector $\mathbf{b} \in R_q^{m_b}$ and a matrix $\mathbf{H}_{\mathcal{L},w,z}$, the following key-homomorphic relation holds:

$$\left(\mathbf{b}^T, \mathbf{a}_u^T - z_u(X)\mathbf{g}^T\right)\mathbf{H}_{\mathcal{L},w,z} = \mathbf{a}_w^T - z_w(X)\mathbf{g}^T,$$

where $\mathbf{a}_w$ is the output of $\mathsf{EvalLut}$ and $\mathbf{H}_{\mathcal{L},w,z}$ is the output of $\mathsf{EvalLutX}$. Notably the matrix satisfies the bound:

$$\left\|\mathbf{H}_{\mathcal{L},w,z}\right\|_\infty \leq \sigma_b\sqrt{\lambda},$$

such that the error term of the resulting encoding after the lookup table evaluation is independent of the table size R and the input norm $|z_u(X)|$.

The homomorphic evaluation of the lookup table is split into two algorithms: $\mathsf{EvalLut}$, a data-independent setup algorithm, and $\mathsf{EvalLutX}$, a data-dependent evaluation algorithm.

- $\mathsf{EvalLut}$. For a given LUT $\mathcal{L} = \{(\bar{x}_k(X), \bar{y}_k(X))\}_{k\in[R]}$, it prepares a set of candidate evaluation matrices, $\{\mathbf{K}_{w,k}\}_{k\in[R]}$, one for each possible entry in the table. For each entry k, it uses the public vector $\mathbf{b}$ and its trapdoor $\mathbf{b}^{-1}$ to generate the matrix $\mathbf{K}_{w,k}$. This process can be parallelized across all entries in the table. Additionally, it produces a freshly sampled public key $\mathbf{a}_w$ for the output encoding.
- $\mathsf{EvalLutX}$ This algorithm performs the actual evaluation dependent on the input polynomial $z_u(X)$. Its working is straightforward: it looks for an index k in the LUT such that $\bar{x}_k(X) = z_u(X)$. If a match is found, it selects the corresponding pre-computed matrix $\mathbf{K}_{w,k}$ from the set generated by $\mathsf{EvalLut}$ and outputs it as the final evaluation matrix $\mathbf{H}_{\mathcal{L},w,z}$.

Informally, security guarantees that choosing an index k' such that the input polynomial $z_u(X)$ does not match the table entry $\bar{x}_{k'}(X)$ results in a corrupted output. Specifically, the key-homomorphic relation produces an additional non-zero term $(\bar{x}_{k'}(X) - z_u(X))\mathbf{g}^T\mathbf{K}_{w,k',\mathsf{low}}$. Since $\mathbf{K}_{w,k',\mathsf{low}}$ is a matrix with entries sampled from a discrete Gaussian distribution, this term acts as a large, structured noise component added to the result. This effectively randomizes the output.

Having introduced the lookup table evaluation gate, we can now describe its application to achieve fully-domain multiplication over BGG+ encodings.

2.3 Fully-domain arithmetic over BGG+.

The key-homomorphic multiplication between two BGG+ encodings is only guaranteed when the left input carried into a multiplication gate is norm-bounded.

Indeed, in the last sentence of the BGG+ paper, the authors mention that "it would also be useful to construct an efficient ABE for arithmetic circuits where multiplication gates can handle inputs as large as the modulus q." Our new lookup table (LUT) evaluation gate directly addresses this challenge. It allows for the homomorphic evaluation of non-linear functions that cannot be expressed via arithmetic circuits, which in turn enables fully-domain arithmetic over BGG+ encodings.

First we set the goal of performing multiplication on BGG+ encodings of two integers $\tilde{a}, \tilde{b} \in \mathbb{Z}_q$ to obtain an encoding of their product $\tilde{a}\tilde{b} \pmod q$. To achieve this, we represent integers using a multi-precision, base-B digit representation. Fix a base $B \geq 2$ and let $w := \lceil \log_B q \rceil$. For an integer $\tilde{a} \in \mathbb{Z}_q$, we write the corresponding base-B digit representation as $\tilde{\mathbf{a}} = (\tilde{a}_1, \ldots, \tilde{a}_w)$ such that

$$\tilde{a} = \sum_{\ell=1}^{w} B^{\ell-1} \tilde{a}_\ell,$$

where each digit $\tilde{a}_\ell \in \mathbb{Z}_q$ has coefficients in $[0, B)$. Since the system we are working with allows to encode polynomials $\in \mathcal{R}_q$, we can easily encode each vector $\tilde{\mathbf{a}}, \tilde{\mathbf{b}} \in \mathbb{Z}_q^w$ via w BGG+ encodings of constant polynomials. However, standard multi-precision modular multiplication algorithms [16] require operations such as floor division and modular reduction by the base B, which cannot be expressed using the operations supported by BGG+ encodings. The only known way to implement these non-linear operations homomorphically is to set the base $B = 2$ and use binary circuits. The significant downside to this approach is the large number of encodings required to represent a single integer, which is $\log_2 q$. Our LUT evaluation gate makes it practical to use a larger base $B > 2$. We introduce a circuit, $C_{\mathsf{Mod\&Floor}}$ which leverages the LUT gate to homomorphically compute the modulus reduction $\tilde{a}_\ell \bmod B$ and the floor division $\left\lfloor \frac{\tilde{a}_\ell}{B} \right\rfloor$. $C_{\mathsf{Mod\&Floor}}$ effectively provides the necessary non-linear operations for multi-precision modular arithmetic on B-radix integers, while guaranteeing that the output is norm-bounded and therefore suitable for input into a multiplication gate. Compared to binary circuits, using a larger base B reduces the number of encodings required to represent an integer to $\log_B q$. Similar logic can be applied to obtain multi-precision modular addition and subtraction.

As of now, each encoding carries a single digit, such that w encodings are necessary to represent an integer. The next goal is to leverage the full power of polynomial encoding by using the entire set of n coefficients and perform modular arithmetic in a SIMD fashion. The initial insight is to leverage the Number Theoretic Transform (NTT) and its inverse to pack n digits into a single polynomial. Inverse NTT takes as input a vector of n integers and outputs a polynomial in $\mathcal{R}_q$

$$\mathsf{NTT}^{-1} : \mathbb{Z}_q^n \to \mathcal{R}_q$$

Additionally, by the ring-homomorphism property of the NTT, all ring operations over $\mathcal{R}_q$ correspond to component wise arithmetics over Z_q^n such that the

following holds:

$$a(X) + b(X) = \mathsf{NTT}^{-1}(\tilde{\mathbf{a}} \boxplus \tilde{\mathbf{b}}),$$
$$a(X) - b(X) = \mathsf{NTT}^{-1}(\tilde{\mathbf{a}} \boxminus \tilde{\mathbf{b}}),$$
$$a(X)b(X) = \mathsf{NTT}^{-1}(\tilde{\mathbf{a}} \boxtimes \tilde{\mathbf{b}}),$$

where $a(X) := \mathsf{NTT}^{-1}(\tilde{\mathbf{a}})$ and $b(X) := \mathsf{NTT}^{-1}(\tilde{\mathbf{b}})$.

The new problem is that, even though the integers are bounded by B, the infinity norm of the resulting polynomial can be as large as q. This prevents us from using these polynomials as inputs to multiplication gates. To address this, we rely on the CRT representation by choosing q as a product of h co-prime odd prime such that $q = \prod_{i=1}^{h} q_i$. Leveraging the CRT representation, we obtain an inverse NTT transformation for each modulus q_i such that $\mathsf{NTT}_i^{-1} : \mathbb{Z}_{q_i}^n \to \mathcal{R}_{q_i}$ for $i \in [h]$. The infinity norm of the resulting polynomial $a_i(X) := \mathsf{NTT}_i^{-1}(\tilde{\mathbf{a}})$ is now bounded by q_i, which allows us to use it as input to multiplication gates. We can now pack n integers $\tilde{a}_1, \ldots, \tilde{a}_n \in \mathbb{Z}_q$ into a matrix of $w \times h$ polynomials $\{a_{\ell,i}(X)\}_{\ell \in [w], i \in [h]}$ such that $a_{\ell,i}(X) := \mathsf{NTT}_i^{-1}(\tilde{a}_{1,i,\ell}, \ldots, \tilde{a}_{j,i,\ell}, \ldots, \tilde{a}_{n,i,\ell})$ where $\tilde{a}_{j,i,\ell}$ is the ℓ-th base-B digit of $\tilde{a}_j \bmod q_i$. Since the ring homomorphic properties of the NTT hold for each modulus q_i, we only need to define the $\mathsf{Mod\&Floor}$ circuit for packed input polynomials and we can then perform fully-domain modular arithmetic in a SIMD fashion.

To achieve this, the $\mathsf{Mod\&Floor}$ circuit operates on an input polynomial $a(X)$ (which encodes n digits $(\tilde{a}_j)_{j \in [n]}$ in its NTT slots) in a SIMD manner. First, for each slot $j \in [n]$, it isolates the digit $\tilde{a}_j$ by multiplying $a(X)$ with the j-th Lagrange basis polynomial, $\ell_{i,j}(X)$. This creates a new polynomial $a'_j(X)$ that only encodes the j-th digit. Then, we construct two lookup tables, $\mathcal{L}_{\mathsf{mod}}$ and $\mathcal{L}_{\mathsf{floor}}$, which map a polynomial encoding a value k in slot j to a polynomial encoding the result ($k \bmod B$ or $\lfloor k/B \rfloor$) in the same slot. By applying these LUTs to each $a'_j(X)$, we obtain n result polynomials. Summing these result polynomials leverages the orthogonality of the Lagrange basis to combine the outcomes from all slots into a single final output polynomial. By composing the native BGG+ homomorphic operations with the $C_{\mathsf{Mod\&Floor}}$ circuit, we can therefore construct key-homomorphic evaluation schemes for any full-domain arithmetic circuit that operates on vectors of integers in $\mathbb{Z}_q^n$.

2.4 Fully-domain arithmetic ABE

3 Preliminaries

3.1 Notations

For any integers $i, j \in \mathbb{Z}$ such that $i \leq j$ holds, let $[0]$, $[i]$, $[i, j]$, respectively, denote sets $\emptyset$, $\{1, \ldots, i\}$, and $\{i, i+1, \ldots, j\}$. We write a vector and a matrix in bold letters, e.g., $\mathbf{a}$ and $\mathbf{A}$. Unless explicitly noted, vectors are column vectors. An entry of a vector $\mathbf{a}$ and a matrix $\mathbf{A}$ are specified by a_i and $a_{i,j}$. Let $(a_i)_{i \in [n]}$

represent a vector with n rows. The size of $\mathbf{a}$ is denoted by $|\mathbf{a}|$. For n elements $a_1, \ldots, a_n$, For a matrix $\mathbf{A}$ with n rows and m columns, the ℓ_2-norm and the entry-wise infinity norm of $\mathbf{A}$ are, respectively, defined as below:

$$\|\mathbf{A}\|_2 := \max_{\|\mathbf{x}\|_2=1} \|\mathbf{A}\mathbf{x}\|_2 ,$$
$$\|\mathbf{A}\|_\infty := \max_{i\in[n],j\in[m]} \{|a_{i,j}|\},$$

where $\|\mathbf{x}\|_2 := \sqrt{\Sigma_{i\in|\mathbf{x}|} x_i^2}$.

We denote by $(\mathbf{A}, \mathbf{B})$ the column-wise concatenation of two matrices $\mathbf{A}$ and $\mathbf{B}$ with n rows and m_1 and m_2 columns, respectively. For a matrix $\mathbf{A} := (\mathbf{a}_1, \ldots, \mathbf{a}_m)$ with n rows and m columns, let $\mathsf{flatten}(\mathbf{A})$ denote a function to flatten the columns of $\mathbf{A}$, i.e., $\mathsf{flatten}(\mathbf{A}) := \begin{pmatrix} \mathbf{a}_1 \\ \vdots \\ \mathbf{a}_m \end{pmatrix}$. For any $m \in \mathbb{N}$ and $i \in [m]$, let $\mathbf{u}_{i,m}$ and $\mathbf{0}_m$ denote the i-th unit vector of size m and the zero vector of size m, respectively. An identity matrix and a zero matrix of size m are denoted by $\mathbf{I}_m$, $\mathbf{0}_m$, respectively.

A security parameter is denoted by λ. A function $\mathsf{negl}(\lambda) : \mathbb{N} \to \mathbb{R}$ is said to be negligible, if for every constant $c > 0$, there exists an integer $n \in \mathbb{N}$ such that $\mathsf{negl}(\lambda) < \lambda^{-c}$ holds for all $\lambda > n$. We use the notation $x \leftarrow\!\!\$\ \mathcal{X}$ to indicate that x is sampled from a distribution $\mathcal{X}$. For two distributions $\mathcal{X}$ and $\mathcal{Y}$, $\mathcal{X} \overset{\mathrm{c}}{\approx} \mathcal{Y}$ means that $\mathcal{X}$ and $\mathcal{Y}$ are computationally indistinguishable for all probabilistic polynomial-time (PPT) adversaries except with a negligible probability. Similarly, $\mathcal{X} \overset{\mathrm{s}}{\approx} \mathcal{Y}$ means that $\mathcal{X}$ and $\mathcal{Y}$ are statistically indistinguishable for all unbounded distinguishers except with a negligible probability.

Let $\mathbb{Z}_q := \mathbb{Z}/q\mathbb{Z}$ be the integers modulo q for the modulus $q \geq 2$. An integer $x \in \mathbb{Z}_q$ has the representative element in $[0, q)$. For two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_q^n$, we write $\mathbf{x} \boxplus \mathbf{y}$, $\mathbf{x} \boxminus \mathbf{y}$, and $\mathbf{x} \boxtimes \mathbf{y}$ for the entrywise addition, subtraction, and multiplication modulo q, respectively; that is, $\mathbf{x} \boxplus \mathbf{y} := (x_i + y_i \bmod q)_{i\in[n]}$, $\mathbf{x} \boxminus \mathbf{y} := (x_i - y_i \bmod q)_{i\in[n]}$, and $\mathbf{x} \boxtimes \mathbf{y} := (x_i y_i \bmod q)_{i\in[n]}$.

We denote a cyclotomic ring by $\mathcal{R} := \mathbb{Z}[X]/(x^n + 1)$, where $n := \mathsf{poly}(\lambda)$ is a power-of-two integer. Let $\mathcal{R}_q := \mathcal{R}/q\mathcal{R}$ be the quotient ring for an integer modulus $q \in \mathbb{N}$. We write $\mathcal{R}_2, \mathcal{R}_3 \subset \mathcal{R}_q$ to denote the sets of polynomials in $\mathcal{R}_q$ such that each coefficient of the polynomial is chosen uniformly at random from $\{0, 1\}$ and $\{0, \pm 1\}$, respectively. A function $\mathsf{Round}_{q/4} : \mathcal{R}_q \to \mathcal{R}_2$ on input a polynomial $a(X) \in \mathcal{R}_q$ outputs another polynomial $b(X) \in \mathcal{R}_2$ such that, for each $i \in [n]$, the i-th coefficient of $b(X)$ is 1 if that of $a(X)$ is within $[\frac{q}{4}, \frac{3q}{4})$ and 0 otherwise.

For a polynomial $a(X) \in \mathcal{R}_q$, let $|a(X)|$ denote the maximum absolute value of the coefficients of $a(X)$. For two matrices $\mathbf{A} \in \mathcal{R}_q^{l\times m}$ and $\mathbf{B} \in \mathcal{R}_q^{m\times \ell}$, it holds that $\|\mathbf{A}\mathbf{B}\|_\infty \leq nm \|\mathbf{A}\|_\infty \|\mathbf{B}\|_\infty$. For a base integer $B \geq 2$ and $m := \lceil \log_B q \rceil$, a digit decomposition of an integer $a \in \mathbb{Z}_q$ is defined by $\mathsf{digits}_{B,m}(a) := (a_1, \ldots, a_m)^T \in \mathbb{Z}_q^m$, where a_i is the i-th least significant digit of a, i.e., $a =$

$\sum_{i\in[m]} B^{i-1}a_i$. Similarly, for a polynomial $a(X) \in \mathcal{R}_q$, we define $\mathsf{digits}_{B,m}(a(X)) := (a_1(X), \ldots, a_m(X))^T \in \mathcal{R}_q^m$, where for every $j \in [n]$, the j-th coefficient of $a_i(X)$ is the i-th least significant digit of the j-th coefficient of $a(X)$, i.e., $a(X) = \sum_{i\in[m]} B^{i-1}a_i(X)$.

3.2 Lattices

Lattices and Gaussian: Fix positive integers $n, m, q \in \mathbb{N}$. For a vector $\mathbf{a} \in \mathcal{R}_q^m$, a q-ary orthogonal lattice is defined by $\Lambda^\perp(\mathbf{a}^T) := \{\mathbf{x} \in \mathcal{R}_q^m : \mathbf{a}^T\mathbf{x} = 0 \mod q\mathcal{R}\}$. Its coset for a polynomial $u(X) \in \mathcal{R}_q$ is defined by $\Lambda^\perp_{u(X)}(\mathbf{a}^T) := \{\mathbf{x} \in \mathcal{R}_q^m : \mathbf{a}^T\mathbf{x} = u(X) \mod q\mathcal{R}\}$.

Let $\mathcal{D}_{\Lambda^\perp_{u(X)}(\mathbf{a}^T),\sigma}$ denote a discrete Gaussian distribution over a lattice coset $\Lambda^\perp_{u(X)}(\mathbf{a}^T)$ for a parameter σ and a center $\mathbf{0}$. For a vector $\mathbf{u} = (u_1(X), \ldots, u_\ell(X))^T \in \mathcal{R}_q^\ell$, let $\mathcal{D}_{\Lambda^\perp_{\mathbf{u}}(\mathbf{a}^T),\sigma} \subset \mathcal{R}_q^{m\times\ell}$ be a distribution such that each column of $\mathbf{K} \leftarrow\!\!\$\ \mathcal{D}_{\Lambda^\perp_{\mathbf{u}}(\mathbf{a}^T),\sigma}$ is sampled from $\mathcal{D}_{\Lambda^\perp_{u_i(X)}(\mathbf{a}^T),\sigma}$ for every $i \in [\ell]$, i.e., $\mathbf{a}^T\mathbf{K} = \mathbf{u}^T$ holds.

Following Lemma 1 in [13], we truncate $\mathcal{D}_{\mathcal{R},\sigma}$ to make the bound deterministic as below.

Lemma 1 (Tail Bound of $\mathcal{D}_{\mathcal{R},\sigma}$ [17, 13]). *There exists $B_0 \in \Theta(\sqrt{\lambda})$ such that*

$$\Pr\left[|e(X)| \geq \sigma B_0 \,:\, e(X) \leftarrow \mathcal{D}_{\mathcal{R},\sigma}\right] \leq 2^{-\lambda} \quad \textit{for all } \sigma \geq 1 \textit{ and } \lambda \in \mathbb{N}.$$

Let $B \geq 0$, the distribution $\mathcal{D}_{\mathcal{R},\sigma,\leq B}$ is sampled by first sampling $e(X) \leftarrow\!\!\$\ \mathcal{D}_{\mathcal{R},\sigma}$, then returning $e(X)$ if $|e(X)| \leq B$, and 0 otherwise. Let $\sigma \geq 1$ and $B = \sigma \cdot \Theta(\sqrt{\lambda})$, then $\mathcal{D}_{\mathcal{R},\sigma,\leq B}$ is $2^{-\Omega(\lambda)}$-close to $\mathcal{D}_{\mathcal{R},\sigma}$.

Gadget matrix: Fix a base $B_g \geq 2$ and $m_g := \left\lceil \log_{B_g} q \right\rceil$. A gadget vector for a base B is defined as follows:

$$\mathbf{g} := (B^0, B^1, \ldots, B^{m_g-1})^T \in \mathcal{R}_q^{m_g}.$$

For a polynomial $u(X) \in \mathcal{R}_q$, a vector $\mathbf{g}^{-1}(u(X)) \in \Lambda^\perp_{u(X)}(\mathbf{g})$ is defined by $\mathbf{g}^{-1}(u(X)) := \mathsf{digits}_{B_g,m_g}(u(X)) \in \mathcal{R}_q^{m_g}$. For a vector $\mathbf{u} \in \mathcal{R}_q^\ell$, a matrix $\mathbf{g}^{-1}(\mathbf{u}) := (\mathbf{g}^{-1}(u_1(X)), \ldots, \mathbf{g}^{-1}(u_\ell(X))) \in \mathcal{R}_q^{m_g\times\ell}$ satisfies $\mathbf{g}^T\mathbf{g}^{-1}(\mathbf{u}) = \mathbf{u}^T$.

Trapdoor and preimage sampling: We use PPT algorithms for trapdoor generation and preimage sampling in the ring setting introduced in prior works:

- $\mathsf{TrapGen}(1^n, 1^m, 1^q) \to (\mathbf{a}, \mathbf{a}_\sigma^{-1})$ [2, 5, 11, 17]: This takes as input integers $n, q \in \mathbb{N}$ and $m = \Theta(\log_2 q)$. This outputs a vector $\mathbf{a} \in \mathcal{R}_q^m$ and a trapdoor $\mathbf{a}_\sigma^{-1}$. The vector $\mathbf{a}$ is $\mathsf{negl}(n)$-close to uniform. For a parameter $\sigma = \mathcal{O}(n \log_2 q)\,\omega(\sqrt{\log_2 m})$ and a vector $\mathbf{u} \in \mathcal{R}_q^\ell$, the trapdoor $\mathbf{a}_\sigma^{-1}$ allows one to efficiently sample a preimage $\mathbf{K} \leftarrow\!\!\$\ \mathbf{a}_\sigma^{-1}(\mathbf{u})$ such that $\mathbf{a}^T\mathbf{K} = \mathbf{u}^T$, $\|\mathbf{K}\|_\infty \leq \sigma\sqrt{\lambda}$, and $\mathbf{K}$ is $\mathsf{negl}(n)$-close to a coset $\mathcal{D}_{\Lambda^\perp_{\mathbf{u}}(\mathbf{a}^T),\sigma} \subset \mathcal{R}_q^{m\times\ell}$.

– $\mathsf{SampLeft}(\mathbf{a}, \mathbf{S}, z(X), \mathbf{u}, \sigma) \to \mathbf{L}$ [1, 7, 8]: This takes as input a vector $\mathbf{a} \in \mathcal{R}_q^m$, a matrix $\mathbf{S} \in \mathcal{R}_q^{m \times \ell}$, a polynomial $z(X) \neq 0$, a vector $\mathbf{u} \in \mathcal{R}_q^\ell$, and a parameter $\sigma > \sqrt{5}\,\|\mathbf{S}\|_2\,\omega(\sqrt{\log_2 m})$. This outputs a vector $\mathbf{L} \in \mathcal{R}_q^{m \times \ell}$ such that $\mathbf{L}$ is $\mathsf{negl}(n)$-close to a coset $\mathcal{D}_{\Lambda_{\mathbf{u}}^\perp((\mathbf{a}^T, \mathbf{a}^T\mathbf{S} + z(X)\mathbf{g}^T)), \sigma}$.[1]

Statistical lemmas: We introduce the following lemmas regarding statistical indistinguishability.

Lemma 2 (Distribution of Preimages [11, 3]). *For* $(\mathbf{a}, \mathbf{T_a}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, 1^q)$, $\sigma = \mathcal{O}(n \log_2 q)\,\omega(\sqrt{\log_2 m})$, *and* $\ell \in \mathbb{N}$, *the following distributions are statistically indistinguishable:*

$$\left\{(\mathbf{a}, \mathbf{K}, \mathbf{u}) : \mathbf{u} \hookleftarrow \mathcal{R}_q^\ell, \mathbf{K} \leftarrow \mathbf{a}_\sigma^{-1}(\mathbf{u})\right\}$$
$$\stackrel{s}{\approx} \left\{(\mathbf{a}, \mathbf{K}, \mathbf{u}) : \mathbf{K} \hookleftarrow \mathcal{D}_{\mathcal{R},\sigma}^{m \times \ell}, \mathbf{u}^T := \mathbf{a}^T\mathbf{K}\right\}.$$

We prove the following lemma about randomness extraction in a ring setting in a similar manner to Lemma 13 in [1].

Lemma 3 (Randomness Extraction [9, 1, 6]). *Let* $1 < d_1, \ldots, d_h \leq n$ *be powers of two. Fix distinct prime integers* $q_1, \ldots, q_h$ *such that* $q_i = 2d_i + 1 \bmod 4d_i$ *for every* $1 \leq i \leq h$, *and* $3 \leq \min_{i \in [h]} \frac{q_i^{1/d_i}}{\sqrt{d_i}}$. *Let positive integers* B *and* $q = \prod_{i \in [h]} q_i$. *For a security parameter* λ, *set*

$$m \geq \frac{1}{\log_2 3}\left(\log_2 q + \log_2 B + \frac{2\lambda}{n}\right). \tag{1}$$

For all vectors $\mathbf{e} \in \mathcal{R}_q^m$ *such that* $|\mathbf{e}| \leq B$ *and* $\mathbf{a} \hookleftarrow \mathcal{R}_q^m$, *it holds that*

$$\left\{(\mathbf{a}, \mathbf{a}^T\mathbf{r},\, \mathbf{e}^T\mathbf{r}) : \mathbf{r} \hookleftarrow \mathcal{R}_3^m\right\} \stackrel{s}{\approx} \left\{(\mathbf{a}, \mathbf{u}^T, \mathbf{e}^T\mathbf{r}) : \mathbf{u} \hookleftarrow \mathcal{R}_q^m\right\}.$$

Proof. By Lemma 3 and Corollary 2 in [6], the function $H_{\mathbf{a},m} : \mathcal{R}_3^m \to \mathcal{R}_q$ *defined by* $H_{\mathbf{a},m}(\mathbf{r}) := \mathbf{a}^T\mathbf{r}$ *is a universal hash function. Therefore, by the generalized leftover hash lemma [9, 1], we obtain*

$$\Delta\left((\mathbf{a}, \mathbf{a}^T\mathbf{r},\, \mathbf{e}^T\mathbf{r}), (\mathbf{a}, \mathbf{u}^T, \mathbf{e}^T\mathbf{r})\right) \leq \frac{1}{2}\sqrt{\frac{(qB)^n}{3^{nm}}}.$$

Inequality 1 implies $\frac{1}{2}\sqrt{\frac{(qB)^n}{3^{nm}}} \leq 2^{-\lambda}$. *Combining these inequalities, we conclude that the statistical distance above is at most* $2^{-\lambda}$. □

Hardness assumptions: Our security proof in Section 6 relies on the hardness of the ring learning with errors (ring-LWE) assumption.

[1] In the original definition in [1], given a trapdoor $\mathbf{b}_\sigma^{-1}$ corresponding to a vector $\mathbf{b}$, one can sample a preimage close to $\mathcal{D}_{\Lambda_{\mathbf{u}}^\perp((\mathbf{a}^T, \mathbf{a}^T\mathbf{S} + \mathbf{b}^T)), \sigma}$. However, in similar to [7], our definition fixes $\mathbf{b}$ to the gadget vector $\mathbf{g}$.

Assumption 1 (Ring Learning with Errors (Ring-LWE) [15]). *Fix positive integers $q \leq 2, m = \mathsf{poly}(\lambda)$ and let n be a power of two. Let χ_s, χ_e be distributions over $\mathcal{R}_q$. The ring-LWE assumption is said to be hard if for every PPT adversary, the following holds:*

$$\begin{aligned}&\left\{(\mathbf{a}, \mathbf{b}) : s(X) \leftarrow\!\$\ \chi_s, \mathbf{e} \leftarrow\!\$\ \chi_e^m, \mathbf{b}^T := s(X)\mathbf{a}^T + \mathbf{e}^T\right\}\\ \overset{c}{\approx}&\left\{(\mathbf{a}, \mathbf{b}) : \mathbf{b} \leftarrow\!\$\ \mathcal{R}_q^m\right\},\end{aligned}$$

where $\mathbf{a} \leftarrow\!\$\ \mathcal{R}_q^m$.

The original definition [15] recommends instantiating the error distribution χ_e as a discrete Gaussian $\mathcal{D}_{\mathcal{R},\sigma}$ with sufficiently large σ. It further suggests taking the secret distribution χ_s to be either the same as the error distribution or uniform over $\mathcal{R}_q$. In practice, however, χ_s is often instantiated with a ternary distribution $\mathcal{R}_3$ to improve efficiency [4]. We therefore use $\mathcal{R}_3$ for χ_s.

3.3 Key-Homomorphic Encoding à la BGG+

We recall key-homomorphic properties of BGG+ encodings [7] with minor modifications to the construction in a ring setting introduced in [8].

Lemma 4 (Key-Homomorphic Properties of BGG+ Encodings [7, 8]). *Fix a positive integer $q, B_g \geq 2$, and let n be a power of two. Set $m_g := \left\lceil \log_{B_g} q \right\rceil$. There exist the following deterministic algorithms that satisfy key-homomorphic properties defined as below.*

- $\mathsf{EvalAdd}(\mathbf{a}_u, \mathbf{a}_v) \to \mathbf{a}_w$*: This takes as input two vectors $\mathbf{a}_u, \mathbf{a}_v \in \mathcal{R}_q^{m_g}$ and outputs a vector $\mathbf{a}_w \in \mathcal{R}_q^{m_g}$.*
- $\mathsf{EvalAddX}() \to \mathbf{H}_{+,w,z}$*: This takes no input and outputs a matrix $\mathbf{H}_{+,w,z} \in \mathcal{R}_q^{2m_g \times m_g}$ such that the following holds:*

$$\begin{aligned}&\left(\mathbf{a}_u^T - z_u(X)\mathbf{g}^T, \mathbf{a}_v^T - z_v(X)\mathbf{g}^T\right)\mathbf{H}_{+,w,z} = \mathbf{a}_w^T - (z_u(X) + z_v(X))\mathbf{g}^T,\\ &\left\|\mathbf{H}_{+,w,z}\right\|_\infty = 1,\end{aligned}$$

  where $\mathbf{a}_w \leftarrow \mathsf{EvalAdd}(\mathbf{a}_u, \mathbf{a}_v)$.
- $\mathsf{EvalMul}(\mathbf{a}_u, \mathbf{a}_v) \to \mathbf{a}_w$*: This takes as input two vectors $\mathbf{a}_u, \mathbf{a}_v \in \mathcal{R}_q^{m_g}$ and outputs a vector $\mathbf{a}_w \in \mathcal{R}_q^{m_g}$.*
- $\mathsf{EvalMulX}(\mathbf{a}_v, z_u(X)) \to \mathbf{H}_{\times,w,z}$*: This takes as input a vectors $\mathbf{a}_v \in \mathcal{R}_q^{m_g}$ and a polynomial $z_u(X) \in \mathcal{R}_q$. This outputs a matrix $\mathbf{H}_{\times,w,z} \in \mathcal{R}_q^{2m_g \times m_g}$ such that the following holds:*

$$\begin{aligned}&\left(\mathbf{a}_u^T - z_u(X)\mathbf{g}^T, \mathbf{a}_v^T - z_v(X)\mathbf{g}^T\right)\mathbf{H}_{\times,w,z} = \mathbf{a}_w^T - (z_u(X)z_v(X))\mathbf{g}^T,\\ &\left\|\mathbf{H}_{\times,w,z}\right\|_\infty \leq \max\left(B - 1, |z_u(X)|\right),\end{aligned}$$

- $\mathsf{EvalScalar}(\mathbf{a}_u, \alpha(X)) \to \mathbf{a}_w$*: This takes as input a vectors $\mathbf{a}_v \in \mathcal{R}_q^{m_g}$ and a polynomial $\alpha(X) \in \mathcal{R}_q$. This outputs a vector $\mathbf{a}_w \in \mathcal{R}_q^{m_g}$.*

– $\mathsf{EvalScalarX}(\alpha(X)) \to \mathbf{H}_{\alpha,w,z}$: *This takes as input a polynomial* $\alpha(X) \in \mathcal{R}_q$. *This outputs a matrix* $\mathbf{H}_{\alpha,w,z} \in \mathcal{R}_q^{m_g \times m_g}$ *such that the following holds:*

$$\left(\mathbf{a}_u^T - z_u(X)\mathbf{g}^T\right)\mathbf{H}_{\alpha,w,z} = \mathbf{a}_w^T - (\alpha(X)z_u(X))\mathbf{g}^T,$$
$$\left\|\mathbf{H}_{\alpha,w,z}\right\|_\infty \leq B-1,$$

Proof. The concrete constructions of these algorithms are provided in [7, 8]. However, to keep the exposition self-contained, we present the constructions in detail below.

– $\mathsf{EvalAdd}(\mathbf{a}_u, \mathbf{a}_v) \to \mathbf{a}_w$: *This outputs* $\mathbf{a}_w^T := \mathbf{a}_u^T + \mathbf{a}_v^T$.
– $\mathsf{EvalAddX}() \to \mathbf{H}_{+,w,z}$; *This outputs* $\mathbf{H}_{+,w,z} := \begin{pmatrix}\mathbf{I}_m \\ \mathbf{I}_m\end{pmatrix}$.
– $\mathsf{EvalMul}(\mathbf{a}_u, \mathbf{a}_v) \to \mathbf{a}_w$: *This outputs* $\mathbf{a}_w^T := \mathbf{a}_u^T\mathbf{g}^{-1}(\mathbf{a}_v)$.
– $\mathsf{EvalMulX}(\mathbf{a}_v, z_u(X)) \to \mathbf{H}_{\times,w,z}$: *This outputs* $\mathbf{H}_{\times,w,z} := \begin{pmatrix}\mathbf{g}^{-1}(\mathbf{a}_v) \\ z_u(X)\mathbf{I}_{m_g}\end{pmatrix}$.
– $\mathsf{EvalScalar}(\mathbf{a}_u, \alpha(X)) \to \mathbf{a}_w$: *This outputs* $\mathbf{a}_w^T := \mathbf{a}_u^T\mathbf{g}^{-1}(\alpha(X)\mathbf{g})$.
– $\mathsf{EvalScalarX}(\alpha(X)) \to \mathbf{H}_{\alpha,w,z}$: *This outputs* $\mathbf{H}_{\alpha,w,z} := \mathbf{g}^{-1}(\alpha(X)\mathbf{g})$.

The outputs of $\mathsf{EvalAdd}$ *and* $\mathsf{EvalAddX}$ *algorithms satisfies required properties since it holds that*

$$\begin{aligned}&\left(\mathbf{a}_u^T - z_u(X)\mathbf{g}^T, \mathbf{a}_v^T - z_v(X)\mathbf{g}^T\right)\mathbf{H}_{+,w,z}\\ =&(\mathbf{a}_u^T - z_u(X)\mathbf{g}^T) + (\mathbf{a}_v^T - z_v(X)\mathbf{g}^T)\\ =&\mathbf{a}_w^T - (z_u(X) + z_v(X))\mathbf{g}^T,\end{aligned}$$

and $\left\|\mathbf{H}_{+,w,z}\right\|_\infty = 1$ *also holds.*

Additionally, the outputs of $\mathsf{EvalMul}$ *and* $\mathsf{EvalMulX}$ *algorithms satisfies required properties since it holds that*

$$\begin{aligned}&\left(\mathbf{a}_u^T - z_u(X)\mathbf{g}^T, \mathbf{a}_v^T - z_v(X)\mathbf{g}^T\right)\mathbf{H}_{\times,w,z}\\ =&(\mathbf{a}_u^T - z_u(X)\mathbf{g}^T)\mathbf{g}^{-1}(\mathbf{a}_v) + z_u(X)(\mathbf{a}_v^T - z_v(X)\mathbf{g}^T)\\ =&\mathbf{a}_w^T - (z_u(X)z_v(X))\mathbf{g}^T,\end{aligned}$$

and $\left\|\mathbf{H}_{\times,w,z}\right\|_\infty \leq \max\left(\left\|\mathbf{g}^{-1}(\mathbf{a}_v)\right\|_\infty, \left\|z_u(X)\mathbf{I}_{m_g}\right\|_\infty\right) \leq \max\left(B-1, |z_u(X)|\right)$ *also holds.*

Finally, the outputs of $\mathsf{EvalScalar}$ *and* $\mathsf{EvalScalarX}$ *algorithms satisfies required properties since it holds that*

$$\begin{aligned}&\left(\mathbf{a}_u^T - z_u(X)\mathbf{g}^T\right)\mathbf{H}_{\alpha,w,z}\\ =&\mathbf{a}_u^T\mathbf{g}^{-1}(\alpha(X)\mathbf{g}) - z_u(X)\mathbf{g}^T\mathbf{g}^{-1}(\alpha(X)\mathbf{g})\\ =&\mathbf{a}_w^T - (\alpha(X)z_u(X))\mathbf{g}^T,\end{aligned}$$

and $\left\|\mathbf{H}_{\alpha,w,z}\right\|_\infty \leq \left\|\mathbf{g}^{-1}(\alpha(X)\mathbf{g})\right\|_\infty \leq B-1$ *also holds.* □

3.4 Double-CRT Representation

In the following, a modulus $q \in \mathbb{N}$ and a degree $n \in \mathbb{N}$ are assumed to be compatible with a double-CRT representation [10]. Specifically, q is a product of h co-prime odd primes, i.e., $q = \prod_{i\in[h]} q_i$. For an integer $a \in \mathbb{Z}_q$ and a polynomial $a(X) \in \mathcal{R}_q$, we write $(a \mod q_i) \in \mathbb{Z}_{q_i}$ and $(a(X) \mod q_i\mathcal{R}) \in \mathcal{R}_{q_i}$ by $[a]_{q_i}$ and $[a(X)]_{q_i}$, respectively. Each q_i satisfies $q_i \equiv 1 \pmod{2n}$, which ensures the existence of a primitive $2n$-th root of unity $[\omega_i]_{q_i} \in \mathbb{Z}_{q_i}$. Let number-theoretic transform (NTT) be defined as below:

$$\begin{aligned}
&\mathsf{NTT}_i : \mathcal{R}_{q_i} \to \mathbb{Z}_{q_i}^n \\
&\mathsf{NTT}_i([a(X)]_{q_i}) := ([a(\omega_i^1)]_{q_i}, [a(\omega_i^3)]_{q_i}, \ldots, [a(\omega_i^{2n-1})]_{q_i})^T.
\end{aligned}$$

The output of $\mathsf{NTT}_i([a(X)]_{q_i})$ is referred to the evaluation slots of $[a(X)]_{q_i}$.

We also define an inverse function of NTT_i as below:

$$\begin{aligned}
&\mathsf{NTT}_i^{-1} : \mathbb{Z}_{q_i}^n \to \mathcal{R}_{q_i} \\
&\mathsf{NTT}_i^{-1}(([a(\omega_i^1)]_{q_i}, [a(\omega_i^3)]_{q_i}, \ldots, [a(\omega_i^{2n-1})]_{q_i})^T) \\
&:= \Sigma_{j\in[n]} \left[a(\omega_i^{2j-1})\right]_{q_i} [\ell_{i,j}(X)]_{q_i} \\
&= [a(X)]_{q_i},
\end{aligned}$$

where $[\ell_{i,j}(X)]_{q_i} := \prod_{k\in[n]/\{j\}} \frac{X-\omega_i^{2k-1}}{\omega_i^{2j-1}-\omega_i^{2k-1}} \mod q_i\mathcal{R}$ is the j-th Lagrange basis polynomial in $\mathcal{R}_{q_i}$ whose domain is $([\omega_i^1]_{q_i}, [\omega_i^3]_{q_i}, [\omega_i^{2n-1}]_{q_i})$. When viewing the output of NTT_i^{-1} as an element of $\mathcal{R}_q$ rather than of $\mathcal{R}_{q_i}$, we write $\mathsf{NTT}_i^{-1}(\mathbf{a}) \mod q\mathcal{R}$.

For an integer $a \in \mathbb{Z}_q$, a decomposition function of integer-CRT (or residue number system) is defined as below:

$$\begin{aligned}
&\mathsf{CRT} : \mathbb{Z}_q \to \mathbb{Z}^h \\
&\mathsf{CRT}(a) := ([a]_{q_1}, \ldots, [a]_{q_h})^T.
\end{aligned}$$

For every $i \in [h]$, let $q_i^\star := q/q_i \in \mathbb{Z}$ and $\tilde{q}_i := q_i^{\star\,-1} \mod q_i \in \mathbb{Z}_{q_i}$. Then for any $\mathbf{a} \in \mathbb{Z}_q^n$ and $a(X) \in \mathcal{R}_q$, as described in [12], it holds that

$$\begin{aligned}
\mathbf{a} &= \boxplus_{i\in[h]}(q_i^\star\tilde{q}_i)([\mathbf{a}]_{q_i}) \mod q, \\
a(X) &= \Sigma_{i\in[h]}(q_i^\star\tilde{q}_i)([a(X)]_{q_i}) \mod q\mathcal{R}.
\end{aligned}$$

By using these properties, the following functions can be defined:

$$\begin{aligned}
&\mathsf{NTT}_{1,\ldots,h} : \mathcal{R}_q \to \mathbb{Z}_q^n \\
&\mathsf{NTT}_{1,\ldots,h}(a(X)) := \boxplus_{i\in[h]}(q_i^\star\tilde{q}_i)\ \left(\mathsf{NTT}_i([a(X)]_{q_i})\right) \mod q, \\
&\mathsf{NTT}_{1,\ldots,h}^{-1} : \mathbb{Z}_q^n \to \mathcal{R}_q \\
&\mathsf{NTT}_{1,\ldots,h}^{-1}(\mathbf{a}) := \Sigma_{i\in[h]}(q_i^\star\tilde{q}_i)\ \mathsf{NTT}_i^{-1}([\mathbf{a}]_{q_i}) \mod q\mathcal{R}.
\end{aligned}$$

Algorithm 1 EvalLut

Input: $\mathbf{a}_u \in \mathcal{R}_q^{m_g}$, trapdoor $\mathbf{b}_{\sigma_b}^{-1}$ for $\mathbf{b} \in \mathcal{R}_q^{m_b}$, $\mathcal{L} \in (\mathcal{R}_q \times \mathcal{R}_q)^R$.
Output: $\mathbf{a}_w \in \mathcal{R}_q^{m_g}$, $\mathbf{K}_{w,1}, \ldots, \mathbf{K}_{w,R} \in \mathcal{R}_q^{(m_b+m_g)\times m_g}$.

$\mathbf{a}_w \hookleftarrow\$\ \mathcal{R}_q^{m_g}$.
Parse $\mathcal{L}$ as $\Big((\bar{x}_1(X), \bar{y}_1(X)), \ldots, (\bar{x}_R(X), \bar{y}_R(X))\Big)$.
for $k \in [R]$ **do**
  $\mathbf{K}_{w,k,\mathsf{low}} \hookleftarrow\$\ \mathcal{D}^{m_g \times m_g}_{\mathcal{R},\sigma_b,\leq\sigma_b\sqrt{\lambda}}$.
  $\mathbf{p}_{w,k}^T := \mathbf{a}_w^T - \bar{y}_k(X)\mathbf{g}^T - \left(\mathbf{a}_u^T - \bar{x}_k(X)\mathbf{g}^T\right)\mathbf{K}_{w,k,\mathsf{low}}$.
  $\mathbf{K}_{w,k,\mathsf{high}} \hookleftarrow\$\ \mathbf{b}_{\sigma_b}^{-1}(\mathbf{p}_{w,k})$.
  $\mathbf{K}_{w,k} := \begin{pmatrix}\mathbf{K}_{w,k,\mathsf{high}} \\ \mathbf{K}_{w,k,\mathsf{low}}\end{pmatrix}$.
end for
return $\mathbf{a}_w$, $\mathbf{K}_{w,1}, \ldots, \mathbf{K}_{w,R}$.

4 Lookup Table Evaluation on BGG+ Encodings

4.1 Key-Homomorphic Evaluation of Lookup Tables

We provide two algorithms to key-homomorphically evaluate a lookup table (LUT), converting a BGG+ encoding and the corresponding public vector for an input polynomial $\bar{x}_k(X) \in \mathcal{R}_q$ into those for an output polynomial $\bar{y}_k(X) \in \mathcal{R}_q$. Notably, these input and output encodings are allowed to encode arbitrary polynomials in $\mathcal{R}_q$, not just bits.

The LUT for polynomials is formally defined as below:

Definition 1 (Lookup Table for Polynomials). *A LUT for polynomials with a size R, denoted by $\mathcal{L}$, is an ordered R-tuple of pairs, each of which consists of a key $\bar{x}_i(X) \in \mathcal{R}_q$ and a value $\bar{y}_i(X) \in \mathcal{R}_q$:*

$$\mathcal{L} := \Big((\bar{x}_1(X), \bar{y}_1(X)), \ldots, (\bar{x}_R(X), \bar{y}_R(X))\Big)^T \in (\mathcal{R}_q \times \mathcal{R}_q)^R.$$

We require inputs to be pairwise distinct: for all $k \neq k' \in [R]$, $\bar{x}_k(X) \neq \bar{x}_{k'}(X)$.

Additionally, for a LUT $\mathcal{L}$, two deterministic functions $\mathsf{LUT}_{\mathcal{L}}^{\mathsf{idx}} : \mathcal{R}_q \to [R] \cup \{\perp\}$ and $\mathsf{LUT}_{\mathcal{L}}^{\mathsf{out}} : \mathcal{R}_q \to \mathcal{R}_q \cup \{\perp\}$ are defined as follows:

$$\mathsf{LUT}_{\mathcal{L}}^{\mathsf{idx}}(z(X)) := \begin{cases} k & \text{if } \exists\, k \in [R] \text{ s.t. } \bar{x}_k(X) = z(X), \\ \perp & \text{otherwise,} \end{cases}$$

$$\mathsf{LUT}_{\mathcal{L}}^{\mathsf{out}}(z(X)) := \begin{cases} \bar{y}_k(X) & \text{if } \exists\, k \in [R] \text{ s.t. } \bar{x}_k(X) = z(X), \\ \perp & \text{otherwise.} \end{cases}$$

The concrete algorithms for evaluating a LUT $\mathcal{L}$ on BGG+ encodings are given in Algorithm 1 and 2. Then these algorithms satisfy the properties stated in the following lemma.

Algorithm 2 $\mathsf{EvalLutX}$

Input: $\mathcal{L} \in (\mathcal{R}_q \times \mathcal{R}_q)^R$, $\mathbf{K}_{w,1}, \ldots, \mathbf{K}_{w,R} \in \mathcal{R}_q^{(m_b+m_g)\times m_g}$, $z_u(X) \in \mathcal{R}_q$.
Output: $\mathbf{H}_{\mathcal{L},w,z} \in \mathcal{R}_q^{(m_b+m_g)\times m_g}$ or $\perp$.
1: $k := \mathsf{LUT}_{\mathcal{L}}^{\mathsf{idx}}(z_u(X))$.
2: **if** $k = \perp$ **then**
3: **return** $\perp$.
4: **end if**
5: $\mathbf{H}_{\mathcal{L},w,z} := \mathbf{K}_{w,k}$.
6: **return** $\mathbf{H}_{\mathcal{L},w,z}$.

Lemma 5 (Key-Homomorphic Evaluation of Lookup Tables). *Fix positive integers $q, B_g \geq 2, R$ and let n be a power of two. Set $m_g := \left\lceil \log_{B_g} q \right\rceil$, $m_b := \Theta(\log_2 q)$ and $\sigma_b := \mathcal{O}(n \log_2 q)\,\omega(\sqrt{\log_2 m_b})$. For any LUT $\mathcal{L} \in (\mathcal{R}_q \times \mathcal{R}_q)^R$ (Definition 1), any vector $\mathbf{a}_u \in \mathcal{R}_q^{m_g}$, and a polynomial $z_u(X) \in \mathcal{R}_q$ such that $\mathsf{LUT}_{\mathcal{L}}^{\mathsf{idx}}(z_u(X)) \neq \perp$, $\mathsf{EvalLut}$ in Algorithm 1 and $\mathsf{EvalLutX}$ in Algorithm 2 satisfy the key-homomorphic property defined as below:*

$$\left(\mathbf{b}^T, \mathbf{a}_u^T - z_u(X)\mathbf{g}^T\right) \mathbf{H}_{\mathcal{L},w,z} = \mathbf{a}_w^T - z_w(X)\mathbf{g}^T, \tag{2}$$

$$\left\|\mathbf{H}_{\mathcal{L},w,z}\right\|_\infty \leq \sigma_b \sqrt{\lambda}, \tag{3}$$

where

$$\begin{aligned}
&(\mathbf{b}, \mathbf{b}_{\sigma_b}^{-1}) \leftarrow \mathsf{TrapGen}(1^n, 1^{m_b}, 1^q),\\
&(\mathbf{a}_w, \mathbf{K}_{w,1}, \ldots, \mathbf{K}_{w,R}) \leftarrow \mathsf{EvalLut}(\mathbf{a}_u, \mathbf{b}_{\sigma_b}^{-1}, \mathcal{L}),\\
&\mathbf{H}_{\mathcal{L},w,z} \leftarrow \mathsf{EvalLutX}(\mathcal{L}, \mathbf{K}_{w,1}, \ldots, \mathbf{K}_{w,R}, z_u(X)),\\
&z_w(X) := \mathsf{LUT}_{\mathcal{L}}^{\mathsf{out}}(z_u(X)).
\end{aligned}$$

Proof. For Eq. 2, let $k := \mathsf{LUT}_{\mathcal{L}}^{\mathsf{idx}}(z_u(X)) \neq \perp$. We have

$$\begin{aligned}
&\left(\mathbf{b}^T, \mathbf{a}_u^T - z_u(X)\mathbf{g}^T\right) \mathbf{H}_{\mathcal{L},w,z}\\
=&\left(\mathbf{b}^T, \mathbf{a}_u^T - z_u(X)\mathbf{g}^T\right) \begin{pmatrix} \mathbf{K}_{w,k,\,\mathsf{high}} \\ \mathbf{K}_{w,k,\,\mathsf{low}} \end{pmatrix}\\
=&\left(\mathbf{a}_w^T - \bar{y}_k(X)\mathbf{g}^T - \left(\mathbf{a}_u^T - \bar{x}_k(X)\mathbf{g}^T\right)\mathbf{K}_{w,k,\,\mathsf{low}}\right) + (\mathbf{a}_u^T - z_u(X)\mathbf{g}^T)\mathbf{K}_{w,k,\,\mathsf{low}}\\
=&\mathbf{a}_w^T - z_w(X)\mathbf{g}^T.
\end{aligned}$$

The second equality follows because by the property of the preimage sampling described in Subsection 3.2, i.e., $\mathbf{b}^T\mathbf{K}_{w,k,\,\mathsf{high}} = \mathbf{p}_{w,k}^T$. The last equation also holds since $\bar{x}_k(X) = z_u(X)$ and $\bar{y}_k(X) = z_w(X)$.

We now prove Inequality 3. Recall that $\mathbf{H}_{\mathcal{L},w,z} = \mathbf{K}_{w,k}$ is a concatenation of $\mathbf{K}_{w,k,\,\mathsf{low}} \hookleftarrow\$\ \mathcal{D}_{\mathcal{R},\sigma_b,\leq\sigma_b\sqrt{\lambda}}^{m_g\times m_g}$ and $\mathbf{K}_{w,k,\,\mathsf{high}} \hookleftarrow\$\ \mathbf{b}_{\sigma_b}^{-1}(\mathbf{p}_{w,k})$. By Lemma 1 and the property of the preimage sampling, it holds that $\left\|\mathbf{K}_{w,k,\,\mathsf{low}}\right\|_\infty \leq \sigma_b\sqrt{\lambda}$ and

$\left\|\mathbf{K}_{w,k,\mathsf{high}}\right\|_\infty \leq \sigma_b\sqrt{\lambda}$, *respectively. Hence, we can conclude that* $\left\|\mathbf{K}_{w,k}\right\|_\infty \leq \sigma_b\sqrt{\lambda}$. □

4.2 Key-Homomorphic Evaluation of Norm-Bounded Circuits

We extend the above key-homomorphic evaluation technique to circuits that evaluate arithmetic operations on polynomials in $\mathcal{R}_q$ along with LUTs, where the norm of any polynomial carried by a left-hand input to a multiplication gate is bounded.

Definition 2 (Norm-Bounded Circuit with Lookup Tables). *Fix a positive integer $q \geq 2$, and let n be a power of two. Let a class $\mathcal{C}_{\mathcal{R}_q,L,M}$ denote the set of circuits $C : \mathcal{R}_q^L \to \mathcal{R}_q^M$ that involve lookup tables. For a polynomial-sized circuit $C \in \mathcal{C}_{\mathcal{R}_q,L,M}$, let N and D denote the number of gates and the depth of C, respectively. Given inputs $\mathbf{x}_L := (x_i(X))_{i\in[L]}$, each wire $u \in [L+N]$ carries a polynomial $z_u(X) \in \mathcal{R}_q$ derived from the inputs, and each gate is assigned an output wire $w \in [N]$ and is of one of the following gate types:*

- *Addition: given $z_u(X), z_v(X) \in \mathcal{R}_q$, output $z_w(X) := z_u(X) + z_v(X) \in \mathcal{R}_q$.*
- *Multiplication: given $z_u(X), z_v(X) \in \mathcal{R}_q$, output $z_w(X) := z_u(X)z_v(X) \in \mathcal{R}_q$.*
- *Scalar multiplication for a fixed polynomial $\alpha(X) \in \mathcal{R}_q$: given $z_u(X) \in \mathcal{R}_q$, output $z_w(X) := \alpha(X)z_u(X) \in \mathcal{R}_q$.*
- *LUT evaluation for a LUT $\mathcal{L}_w \in (\mathcal{R}_q \times \mathcal{R}_q)^{R_w}$, where R_w denotes the table size: given $z_u(X) \in \mathcal{R}_q$, output $z_w(X) := \mathsf{LUT}^{\mathsf{out}}_{\mathcal{L}_w}(z_u(X))$ if $\mathsf{LUT}^{\mathsf{idx}}_{\mathcal{L}_w}(z_u(X)) \neq \perp$, and output $\perp$ otherwise.*

Its outputs $C(\mathbf{x}_L)$ are polynomials $\mathbf{y}_M := (y_i(X))_{i\in[M]} \in \mathcal{R}_q^M$ carried by the output wires in the D-th layer if no LUT evaluation gate outputs $\perp$; otherwise $C(\mathbf{x}_L) = \perp$.

Let $\mathcal{U}_{C,\times} \subset [L+N]$ be the set of all wires that are left-hand inputs to multiplication gates in C. A subset $\mathcal{X}_{C,B_\times} \subseteq \mathcal{R}_q^L$, parameterized by a positive integer $B_\times \geq 2$, is said to be an admissible input set if the following holds:

$$\forall \mathbf{x}_L \in \mathcal{X}_{C,B_\times} \quad (C(\mathbf{x}_L) \neq \perp) \wedge (\forall u \in \mathcal{U}_{C,\times}\ |z_u(X)| \leq B_\times)\,,$$

where $z_u(X) \in \mathcal{R}_q$ is a polynomial carried by a wire u during the evaluation of $C(\mathbf{x}_L)$.

We call multiplication, scalar multiplication, and LUT evaluation gates as non-free gates. For a circuit $C \in \mathcal{C}_{\mathcal{R}_q,L,M}$, its non-free depth D_{NF} is the depth of C after ignoring all addition gates. The addition width of C is the least integer W_+ such that any input wire of every non-free gate or any circuit output wire can be written as a sum of at most W_+ wires, each either the output of a non-free gate or a circuit input wire.

By combining algorithms described in Lemmas 4 and 5, we can define algorithms that key-homomorphically evaluate a polynomial circuit with lookup tables on BGG+ encodings as proven below.

Theorem 1 (Key-Homomorphic Evaluation of Polynomial Circuits). *Fix positive integers $q, B_g \geq 2$ and let n be a power of two. Set $m_g := \left\lceil \log_{B_g} q \right\rceil$, $m_b := \Theta(\log_2 q)$ and $\sigma_b := \mathcal{O}(n \log_2 q)\, \omega(\sqrt{\log_2 m_b})$. For a class $\mathcal{C}_{\mathcal{R}_q,L,M}$ (Definition 2), there exist two deterministic algorithms that satisfy key-homomorphic properties defined as below.*

- $\mathsf{EvalC}_{\mathcal{R}_q}((\mathbf{a}_{\mathsf{inp},i})_{i\in[L]}^T, \mathbf{b}_{\sigma_b}^{-1}, C) \to ((\mathbf{a}_{\mathsf{out},i})_{i\in[M]}^T, \mathcal{K}_{C,\mathsf{LUT}})$*: This takes as input a matrix $(\mathbf{a}_{\mathsf{inp},i})_{i\in[L]}^T \in \mathcal{R}_q^{m_g\times L}$, a trapdoor $\mathbf{b}_{\sigma_b}^{-1}$, and a circuit $C \in \mathcal{C}_{\mathcal{R}_q,L,M}$. This outputs a matrix $(\mathbf{a}_{\mathsf{out},i})_{i\in[M]}^T \in \mathcal{R}_q^{m_g\times M}$ and a set of matrices $\mathcal{K}_{C,\mathsf{LUT}} \subset \bigcup_{R\in[R_{\max}]} \mathcal{R}_q^{(m_b+m_g)R\times m_g}$, where $R_{\max}$ is the maximum lookup table size in C.*
- $\mathsf{EvalCX}_{\mathcal{R}_q}((\mathbf{a}_{\mathsf{inp},i})_{i\in[L]}^T, C, \mathcal{K}_{C,\mathsf{LUT}}, \mathbf{x}_L) \to \mathbf{H}_{C,x}$*: This takes as input a matrix $(\mathbf{a}_{\mathsf{inp},i})_{i\in[L]}^T \in \mathcal{R}_q^{m_g\times L}$, a circuit $C \in \mathcal{C}_{\mathcal{R}_q,L,M}$, a set of matrices $\mathcal{K}_{C,\mathsf{LUT}} \subset \bigcup_{R\in[R_{\max}]} \mathcal{R}_q^{(m_b+m_g)R\times m_g}$, and a vector $\mathbf{x}_L \in \mathcal{R}_q^L$. This outputs a matrix $\mathbf{H}_{C,x} \in \mathcal{R}_q^{(m_b+m_gL)\times m_gM}$ if $C(\mathbf{x}_L) \neq \perp$ and $\perp$ otherwise.*

For any matrix $(\mathbf{a}_{\mathsf{inp},i})_{i\in[L]}^T \in \mathcal{R}_q^{m_g\times L}$, any circuit $C \in \mathcal{C}_{\mathcal{R}_q,L,M}$, and any vector $\mathbf{x}_L := (x_i(X))_{i\in[L]} \in \mathcal{X}_{C,B_\times}$, it holds that

$$\begin{aligned}&\left(\mathbf{b}^T, \mathbf{a}_{\mathsf{inp},1}^T - x_1(X)\mathbf{g}^T, \ldots, \mathbf{a}_{\mathsf{inp},L}^T - x_L(X)\mathbf{g}^T\right)\mathbf{H}_{C,x}\\ &= \left(\mathbf{a}_{\mathsf{out},1}^T - y_1(X)\mathbf{g}^T, \ldots, \mathbf{a}_{\mathsf{out},M}^T - y_M(X)\mathbf{g}^T\right),\end{aligned} \tag{4}$$

where

$$\begin{aligned}&(\mathbf{b}, \mathbf{b}_{\sigma_b}^{-1}) \leftarrow \mathsf{TrapGen}(1^n, 1^{m_b}, 1^g),\\ &((\mathbf{a}_{\mathsf{out},i})_{i\in[M]}^T, \mathcal{K}_{C,\mathsf{LUT}}) \leftarrow \mathsf{EvalC}_{\mathcal{R}_q}((\mathbf{a}_{\mathsf{inp},i})_{i\in[L]}^T, \mathbf{b}_{\sigma_b}^{-1}, C),\\ &\mathbf{H}_{C,x} \leftarrow \mathsf{EvalCX}_{\mathcal{R}_q}((\mathbf{a}_{\mathsf{inp},i})_{i\in[L]}, C, \mathcal{K}_{C,\mathsf{LUT}}, \mathbf{x}_L),\\ &(y_i(X))_{i\in[M]} := C(\mathbf{x}_L).\end{aligned}$$

Additionally, it holds that

$$\left\|\mathbf{H}_{C,X}\right\|_\infty \leq W_+ \left(nB_{\max}W_+(m_b + 2m_g)\right)^{D_{\mathsf{NF}}}, \tag{5}$$

where $B_{\max} := \max(B-1, B_\times, \sigma_b\sqrt{\lambda})$ and D_{NF} is the non-free depth of C.

Proof. *Let D denote the number of layers in the circuit C. Let N_d and $w_{d,i}$, respectively, represent the number of gates and the output wire of the i-th gate in the d-th layer, and $z_{w_{d,i}}$ is a polynomial carried by the wire $w_{d,i}$ during the evaluation of $C(\mathbf{x}_L)$. The $\mathsf{EvalC}_{\mathcal{R}_q}$ (resp. $\mathsf{EvalCX}_{\mathcal{R}_q}$) algorithm is defined by producing $(\mathbf{a}_{w_{d,i}})_{i\in[N_d]}^T \in \mathcal{R}_q^{m_g\times N_d}$ and $\mathcal{K}_{C,\mathsf{LUT}} \subset \bigcup_{R\in[R_{\max}]} \mathcal{R}_q^{(m_b+m_g)R\times m_g}$ (resp. $\mathbf{H}_{C,x,d} \in \mathcal{R}_q^{(m_b+m_gL)\times m_gN_d}$) for every $d \in [D]$—successively from the first up to the D-th layer—that are deterministically derived from the inputs to the*

each algorithm, respectively. We claim that the following relations hold for every $d \in [D]$*:*

$$\begin{aligned}&\left(\mathbf{b}^T, \mathbf{a}_{\mathsf{inp},1}^T - x_1(X)\mathbf{g}^T, \ldots, \mathbf{a}_{\mathsf{inp},L}^T - x_L(X)\mathbf{g}^T\right)\mathbf{H}_{C,x,d}\\ =&\left(\mathbf{a}_{w_{d,1}}^T - z_{w_{d,1}}(X)\mathbf{g}^T, \ldots, \mathbf{a}_{w_{d,N_d}}^T - z_{w_{d,N_d}}(X)\mathbf{g}^T\right). \end{aligned} \tag{6}$$

Starting with $\mathcal{K}_{C,\mathsf{LUT}} = \emptyset$*, we add the preimage matrices to* $\mathcal{K}_{C,\mathsf{LUT}}$ *layer by layer. The* $\mathsf{EvalC}_{\mathcal{R}_q}$ *(resp.* $\mathsf{EvalCX}_{\mathcal{R}_q}$*) algorithm outputs* $(\mathbf{a}_{w_{D,i}})_{i\in[N_D]}^T$ *and* $\mathcal{KC}, \mathsf{LUT}$ *(resp.* $\mathbf{H}_{C,x,D}$*) after processing the last layer. We prove the above claim by induction on* $d \in [D]$*.*

Base case (*$d = 1$***):** For every* $i \in [N_1]$*, the* i*-th gate* $g_{1,i}$ *in the first layer falls into one of four gate types defined in Definition 2. Let* $u_{1,i}$ *(resp.* $v_{1,i}$*) represents the left-hand input (resp. right-hand input) to* $g_{1,i}$*. We show that the claim holds for all gate types as below:*

1. *The first case is that* $g_{1,i}$ *represents addition in* $\mathcal{R}_q$*. Since* $u_{1,i}$ *and* $v_{1,i}$ *are a part of the input wires, namely* $u_{1,i}, v_{1,j} \in [L]$*, we have*

$$\begin{aligned}&\left(\mathbf{b}^T, \mathbf{a}_{\mathsf{inp},1}^T - x_1(X)\mathbf{g}^T, \ldots, \mathbf{a}_{\mathsf{inp},L}^T - x_L(X)\mathbf{g}^T\right)\mathbf{H}_{u_{1,i},x}\\ =&\mathbf{a}_{u_{1,i}}^T - z_{u_{1,i}}(X)\mathbf{g}^T,\\ &\left(\mathbf{b}^T, \mathbf{a}_{\mathsf{inp},1}^T - x_1(X)\mathbf{g}^T, \ldots, \mathbf{a}_{\mathsf{inp},L}^T - x_L(X)\mathbf{g}^T\right)\mathbf{H}_{v_{1,i},x}\\ =&\mathbf{a}_{v_{1,i}}^T - z_{v_{1,i}}(X)\mathbf{g}^T,\end{aligned}$$

   where

$$\begin{aligned}&z_{u_{1,i}}(X) = x_{u_{1,i}}(X), z_{v_{1,i}}(X) = x_{v_{1,i}}(X),\\ &\mathbf{a}_{u_{1,i}} = \mathbf{a}_{\mathsf{inp},u_{1,i}}, \mathbf{a}_{v_{1,i}} = \mathbf{a}_{\mathsf{inp},v_{1,i}},\\ &\mathbf{H}_{u_{1,i},x} := \begin{pmatrix}\mathbf{0}_{(m_b+m_g(u_{1,i}-1))\times m_g}\\ \mathbf{I}_{m_g\times m_g}\\ \mathbf{0}_{m_g(L-u_{1,i})\times m_g}\end{pmatrix} \in \mathcal{R}_q^{(m_b+m_gL)\times m_g},\\ &\mathbf{H}_{v_{1,i},x} := \begin{pmatrix}\mathbf{0}_{(m_b+m_g(v_{1,i}-1))\times m_g}\\ \mathbf{I}_{m_g\times m_g}\\ \mathbf{0}_{m_g(L-v_{1,i})\times m_g}\end{pmatrix} \in \mathcal{R}_q^{(m_b+m_gL)\times m_g}.\end{aligned}$$

   By Lemma 4, we have

$$\mathbf{a}_{w_{1,i}} \leftarrow \mathsf{EvalAdd}(\mathbf{a}_{u_{1,i}}, \mathbf{a}_{v_{1,i}}),\ \mathbf{H}_{w_{1,i},x} := \left(\mathbf{H}_{u_{1,i},x}\ \mathbf{H}_{u_{1,i},x}\right)\mathbf{H}_{+,w_{1,i},z},$$

   where $\mathbf{H}_{+,w_{1,i},z} \leftarrow \mathsf{EvalAddX}()$*. It holds that*

$$\begin{aligned}&\left(\mathbf{b}^T, \mathbf{a}_{\mathsf{inp},1}^T - x_1(X)\mathbf{g}^T, \ldots, \mathbf{a}_{\mathsf{inp},L}^T - x_L(X)\mathbf{g}^T\right)\mathbf{H}_{w_{1,i},x}\\ =&\left(\mathbf{a}_{u_{1,i}}^T - z_{u_{1,i}}(X)\mathbf{g}^T, \mathbf{a}_{v_{1,i}}^T - z_{v_{1,i}}(X)\mathbf{g}^T\right)\mathbf{H}_{+,w_{1,i},z}\\ =&\mathbf{a}_{w_{1,i}}^T - (z_{u_{1,i}}(X) + z_{v_{1,i}}(X))\mathbf{g}^T.\end{aligned}$$

2. *The second case is that $g_{1,i}$ represents multiplication in $\mathcal{R}_q$. In similar to the first case, we can define*

$$\mathbf{a}_{w_{1,i}} \leftarrow \mathsf{EvalMul}(\mathbf{a}_{u_{1,i}}, \mathbf{a}_{v_{1,i}}),\ \mathbf{H}_{w_{1,i},x} := \left(\mathbf{H}_{u_{1,i},x}\ \mathbf{H}_{u_{1,i},x}\right)\mathbf{H}_{\times,w_{1,i},z},$$

where $\mathbf{H}_{\times,w_{1,i},z} \leftarrow \mathsf{EvalMulX}(\mathbf{a}_{v_{1,i}}, z_{u_{1,i}}(X))$. It holds that

$$\begin{aligned}&\left(\mathbf{b}^T, \mathbf{a}_{\mathsf{inp},1}^T - x_1(X)\mathbf{g}^T, \ldots, \mathbf{a}_{\mathsf{inp},L}^T - x_L(X)\mathbf{g}^T\right)\mathbf{H}_{w_{1,i},x}\\ =&\mathbf{a}_{w_{1,i}}^T - (z_{u_{1,i}}(X)z_{v_{1,i}}(X))\mathbf{g}^T.\end{aligned}$$

3. *The third case is that $g_{1,i}$ represents scalar multiplication by $\alpha(X) \in \mathcal{R}_q$. Since it takes only one input wire $u_{1,i}$, we obtain*

$$\mathbf{a}_{w_{1,i}} \leftarrow \mathsf{EvalScalar}(\mathbf{a}_{u_{1,i}}, \alpha(X)),\ \mathbf{H}_{w_{1,i},x} := \mathbf{H}_{u_{1,i},x}\mathbf{H}_{\alpha,w_{1,i},z},$$

where $\mathbf{H}_{\alpha,w_{1,i},z} \leftarrow \mathsf{EvalScalarX}(\alpha(X))$. It holds that

$$\begin{aligned}&\left(\mathbf{b}^T, \mathbf{a}_{\mathsf{inp},1}^T - x_1(X)\mathbf{g}^T, \ldots, \mathbf{a}_{\mathsf{inp},L}^T - x_L(X)\mathbf{g}^T\right)\mathbf{H}_{w_{1,i},x}\\ =&\mathbf{a}_{w_{1,i}}^T - (\alpha(X)z_{u_{1,i}}(X))\mathbf{g}^T.\end{aligned}$$

4. *The last case is that $g_{1,i}$ represents LUT evaluation for a LUT $\mathcal{L}_{w_{1,i}} \in (\mathcal{R}_q \times \mathcal{R}_q)^{R_{w_{1,i}}}$. Recall that $\mathbf{x}_L$ is assumed to be in the admissible input set $\mathcal{X}_{C,B_\times}$ (Definition 2); thus we have $\mathsf{LUT}^{\mathsf{out}}_{\mathcal{L}_{w_{1,i}}}(z_{u_{1,i}}(X)) \neq \perp$. Hence, we can define*

$$\begin{aligned}&(\mathbf{a}_{w_{1,i}}, \mathbf{K}_{w_{1,i},1}, \ldots, \mathbf{K}_{w_{1,i},R_{w_{1,i}}}) \leftarrow \mathsf{EvalLut}(\mathbf{a}_{u_{1,i}}, \mathbf{b}_{\sigma_b}^{-1}, \mathcal{L}_{w_{1,i}}),\\ &\mathbf{H}_{w_{1,i},x} := \left(\mathbf{H}_b\ \mathbf{H}_{u_{1,i},x}\ \mathbf{H}_{u_{1,i},x}\right)\mathbf{H}_{\mathcal{L},w_{1,i},z},\end{aligned}$$

where $\mathbf{H}_{\mathcal{L},w_{1,i},z} \leftarrow \mathsf{EvalLutX}(\mathcal{L}_{w_{1,i}}, \mathbf{K}_{w_{1,i},1}, \ldots, \mathbf{K}_{w_{1,i},R_{w_{1,i}}})$ and $\mathbf{H}_b := \begin{pmatrix}\mathbf{I}_{m_b}\\ \mathbf{0}_{m_gL\times m_b}\end{pmatrix}$. It holds that

$$\begin{aligned}&\left(\mathbf{b}^T, \mathbf{a}_{\mathsf{inp},1}^T - x_1(X)\mathbf{g}^T, \ldots, \mathbf{a}_{\mathsf{inp},L}^T - x_L(X)\mathbf{g}^T\right)\mathbf{H}_{w_{1,i},x}\\ =&\mathbf{a}_{w_{1,i}}^T - \mathsf{LUT}^{\mathsf{out}}_{\mathcal{L}_{w_{1,i}}}(z_{u_{1,i}}(X))\mathbf{g}^T.\end{aligned}$$

We update $\mathcal{K}_{C,\mathsf{LUT}}$ to $\mathcal{K}_{C,\mathsf{LUT}} \leftarrow \mathcal{K}_{C,\mathsf{LUT}} \cup \{\mathbf{K}_{w_{1,i},1}, \ldots, \mathbf{K}_{w_{1,i},R_{w_{1,i}}}\}$.

It follows that $\mathbf{H}_{C,x,1} := \left(\mathbf{H}_{w_{1,1},x} \ldots \mathbf{H}_{w_{1,N_1},x}\right)$ satisfies Eq. 6.

***Inductive step** ($d \in [2, D]$): For every $d' \in [d-1]$, assume that there exist matrices $(\mathbf{a}_{w_{d',i}})_{i\in[N_{d'}]}^T \in \mathcal{R}_q^{m_g\times N_{d'}}$ and $\mathbf{H}_{C,x,d'} \in \mathcal{R}_q^{(m_b+m_gL)\times m_gN_{d'}}$ such that Eq. 6 holds. We show that the same holds for $(\mathbf{a}_{w_{d,i}})_{i\in[N_d]}^T \in \mathcal{R}_q^{m_g\times N_d}$ and $\mathbf{H}_{C,x,d} \in \mathcal{R}_q^{(m_b+m_gL)\times m_gN_d}$.*

For every $d' \in [d-1]$, we can parse $\mathbf{H}_{C,x,d'}$ as below:

$$\mathbf{H}_{C,x,d'} := \left(\mathbf{H}_{w_{d',1},x} \ldots \mathbf{H}_{w_{d',N_{d'}},x}\right),$$

By the inductive hypothesis, for every $i \in [N_{d'}]$, it holds that

$$\begin{aligned}&\left(\mathbf{b}^T, \mathbf{a}_{\mathsf{inp},1}^T - x_1(X)\mathbf{g}^T, \ldots, \mathbf{a}_{\mathsf{inp},L}^T - x_L(X)\mathbf{g}^T\right)\mathbf{H}_{w_{d',i},x}\\ =&\mathbf{a}_{w_{d',i}}^T - z_{w_{d',i}}(X)\mathbf{g}^T. \end{aligned} \tag{7}$$

For every $i \in [N_d]$, let $u_{d,i}$ (resp. $v_{d,i}$) represents the left-hand input (resp. right-hand input) to $g_{d,i}$. Since $u_{d,i}$ and $v_{d,i}$ are the output wires of the gates in some layer d' with $d' \in [d-1]$, we have $(\mathbf{a}_{u_{d,i}}, \mathbf{H}_{u_{d,i},x})$ and $(\mathbf{a}_{v_{d,i}}, \mathbf{H}_{v_{d,i},x})$, each of which satisfies Eq. 7. For each gate type of $g_{d,i}$, we can define $\mathbf{a}_{w_{d,i}}$ and $\mathbf{H}_{w_{d,i},x}$ as follows:

1. *If $g_{d,i}$ is for addition in $\mathcal{R}_q$, we define*

$$\mathbf{a}_{w_{d,i}} \leftarrow \mathsf{EvalAdd}(\mathbf{a}_{u_{d,i}}, \mathbf{a}_{v_{d,i}}),\ \mathbf{H}_{w_{d,i},x} := \left(\mathbf{H}_{u_{d,i},x}\ \mathbf{H}_{u_{d,i},x}\right)\mathbf{H}_{+,w_{d,i},z},$$

   where $\mathbf{H}_{+,w_{d,i},z} \leftarrow \mathsf{EvalAddX}()$.
2. *If $g_{d,i}$ is for multiplication in $\mathcal{R}_q$, we define*

$$\mathbf{a}_{w_{d,i}} \leftarrow \mathsf{EvalMul}(\mathbf{a}_{u_{d,i}}, \mathbf{a}_{v_{d,i}}),\ \mathbf{H}_{w_{d,i},x} := \left(\mathbf{H}_{u_{d,i},x}\ \mathbf{H}_{u_{d,i},x}\right)\mathbf{H}_{\times,w_{d,i},z},$$

   where $\mathbf{H}_{\times,w_{d,i},z} \leftarrow \mathsf{EvalMulX}(\mathbf{a}_{v_{d,i}}, z_{u_{d,i}}(X))$.
3. *If $g_{d,i}$ is for scalar multiplication by $\alpha(X) \in \mathcal{R}_q$, we define*

$$\mathbf{a}_{w_{d,i}} \leftarrow \mathsf{EvalScalar}(\mathbf{a}_{u_{d,i}}, \alpha(X)),\ \mathbf{H}_{w_{d,i},x} := \mathbf{H}_{u_{d,i},x}\mathbf{H}_{\alpha,w_{d,i},z},$$

   where $\mathbf{H}_{\alpha,w_{d,i},z} \leftarrow \mathsf{EvalScalarX}(\alpha(X))$.
4. *If $g_{d,i}$ performs LUT evaluation for a LUT $\mathcal{L}_{w_{d,i}} \in (\mathcal{R}_q \times \mathcal{R}_q)^{R_{w_{d,i}}}$, we define*

$$\begin{aligned}&(\mathbf{a}_{w_{d,i}}, \mathbf{K}_{w_{d,i},1}, \ldots, \mathbf{K}_{w_{d,i},R_{w_{d,i}}}) \leftarrow \mathsf{EvalLut}(\mathbf{a}_{u_{d,i}}, \mathbf{b}_{\sigma_b}^{-1}, \mathcal{L}_{w_{d,i}}),\\ &\mathbf{H}_{w_{d,i},x} := \left(\mathbf{H}_b\ \mathbf{H}_{u_{d,i},x}\ \mathbf{H}_{u_{d,i},x}\right)\mathbf{H}_{\mathcal{L},w_{d,i},z},\end{aligned}$$

   where $\mathbf{H}_{\mathcal{L},w_{d,i},z} \leftarrow \mathsf{EvalLutX}(\mathcal{L}_{w_{d,i}}, \mathbf{K}_{w_{d,i},1}, \ldots, \mathbf{K}_{w_{d,i},R_{w_{d,i}}})$. We update $\mathcal{K}_{C,\mathsf{LUT}}$ to $\mathcal{K}_{C,\mathsf{LUT}} \leftarrow \mathcal{K}_{C,\mathsf{LUT}} \cup \{\mathbf{K}_{w_{d,i},1}, \ldots, \mathbf{K}_{w_{d,i},R_{w_{d,i}}}\}$.

In the same manner as the base case, we can prove that

$$\mathbf{H}_{C,x,d} := \left(\mathbf{H}_{w_{d,1},x} \ldots \mathbf{H}_{w_{d,N_d},x}\right)$$

satisfies Eq. 6. Therefore, by induction, Eq. 6 holds for all $d \in [D]$.

We next confirm the bound of $\left\|\mathbf{H}_{C,X}\right\|_\infty$. Recall that D_{NF} denotes the non-free depth of C as defined in Definition 2. For any non-free gate g in C, we say that g is in the d_{NF}-th non-free layer if the number of non-free gates on any path from the circuit input wires to any input of g (ignoring addition gates) is at most $d_{\mathsf{NF}} - 1$. Consequently, $1 \le d_{\mathsf{NF}} \le D_{\mathsf{NF}}$ holds.

Recall that W_+ denotes the addition width of C as defined in Definition 2. The input of any gate in the first non-free layer is a sum of at most W_+ input

wires only (no prior outputs of non-free gates). Therefore, for any output wire of the i-th gate in the first non-free layer, denoted by $w_{1,\mathsf{NF},i}$, it holds that

$$\begin{aligned}
&\left\|\mathbf{H}_{w_{1,\mathsf{NF},i},x}\right\|_\infty \\
\leq& n(m_b+2m_g)\left\|\left(\mathbf{H}_b\ \mathbf{H}_{u_{1,\mathsf{NF},i},x}\ \mathbf{H}_{v_{1,\mathsf{NF},i},x}\right)\right\|_\infty B_{\max} \\
\leq& nB_{\max}W_+(m_b+2m_g),
\end{aligned}$$

where $u_{1,\mathsf{NF},i}$ and $v_{1,\mathsf{NF},i}$ are the inputs of the above i-th gate. For $d_{\mathsf{NF}}>1$, we have

$$\begin{aligned}
&\left\|\mathbf{H}_{w_{d_{\mathsf{NF}},\mathsf{NF},i},x}\right\|_\infty \\
\leq& n(m_b+2m_g)\left\|\left(\mathbf{H}_b\ \mathbf{H}_{u_{d_{\mathsf{NF}},\mathsf{NF},i},x}\ \mathbf{H}_{v_{d_{\mathsf{NF}},\mathsf{NF},i},x}\right)\right\|_\infty B_{\max} \\
\leq& nB_{\max}\left(m_b+2m_g\right)\Sigma_{j\in[W_+]}\left(\left\|\mathbf{H}_{w_{d_{\mathsf{NF}}-1,\mathsf{NF},j},x}\right\|_\infty\right) \\
\leq& nB_{\max}\left(m_b+2m_g\right)W_+\left(nB_{\max}W_+(m_b+2m_g)\right)^{d_{\mathsf{NF}}-1} \\
=& \left(nB_{\max}W_+(m_b+2m_g)\right)^{d_{\mathsf{NF}}}.
\end{aligned}$$

Since any output wire of C is a sum of at most W_+ outputs of non-free gates, it follows that $\left\|\mathbf{H}_{C,x}\right\|_\infty \leq W_+\left(nB_{\max}W_+(m_b+2m_g)\right)^{D_{\mathsf{NF}}}$. We can conclude the outputs of the $\mathsf{EvalC}_{\mathcal{R}_q}$ and $\mathsf{EvalCX}_{\mathcal{R}_q}$ algorithms—namely $(\mathbf{a}_{\mathsf{out},i})_{i\in[M]}^T = (\mathbf{a}_{w_{d,i}})_{i\in[N_D]}^T$ and $\mathbf{H}_{C,x}=\mathbf{H}_{C,x,D}$—satisfies Eq. 4 and Inequality 5. □

5 Full-domain Arithmetic on BGG+ Encodings

Definition 3 (Full-Domain Vector Arithmetic Circuit). *Fix a positive integer $q\geq 2$, and let n be a power of two. Let a class $\mathcal{C}_{\mathbb{Z}_q^n,L,M}$ denote the set of circuits $C:\mathbb{Z}_q^{n\times L}\to\mathbb{Z}_q^{n\times M}$. For a polynomial-sized circuit $C\in\mathcal{C}_{\mathbb{Z}_q^n,L,M}$, let N and D denote the number of gates and the depth of C, respectively. Given inputs $(\tilde{\mathbf{x}}_i)_{i\in[L]}^T$, each wire $u\in[L+N]$ carries a vector $\tilde{\mathbf{z}}_u\in\mathbb{Z}_q^n$ derived from the inputs, and each gate is assigned an output wire $w\in[N]$ and is of one of the following gate types:*

- *Addition: given $\tilde{\mathbf{z}}_u,\tilde{\mathbf{z}}_v\in\mathbb{Z}_q^n$, output $\tilde{\mathbf{z}}_w:=\tilde{\mathbf{z}}_u\boxplus\tilde{\mathbf{z}}_v\in\mathbb{Z}_q^n$.*
- *Subtraction: given $\tilde{\mathbf{z}}_u,\tilde{\mathbf{z}}_v\in\mathbb{Z}_q^n$, output $\tilde{\mathbf{z}}_w:=\tilde{\mathbf{z}}_u\boxminus\tilde{\mathbf{z}}_v\in\mathbb{Z}_q^n$.*
- *Multiplication: given $\tilde{\mathbf{z}}_u,\tilde{\mathbf{z}}_v\in\mathbb{Z}_q^n$, output $\tilde{\mathbf{z}}_w:=\tilde{\mathbf{z}}_u\boxtimes\tilde{\mathbf{z}}_v\in\mathbb{Z}_q^n$.*

Its outputs $C((\tilde{\mathbf{x}}_i)_{i\in[L]}^T)$ are a matrix $(\mathbf{y}_i)_{i\in[M]}^T\in\mathbb{Z}_q^{n\times M}$ carried by the output wires in the D-th layer.

Lemma 6. *We keep the same parameter settings in Definition 4. For every $i\in[h]$, a circuit $C_{\mathsf{Mod\&Floor},i}$ implementing Algorithm 3 belongs to the class $\mathcal{C}_{\mathcal{R}_q,1,2}$ and satisfies the following properties:*

Algorithm 3 $\mathsf{Mod\&Floor}_{q,n,q_i,B_p}$

Parameter Settings: $q_i | q$ and $(B_p - 1)^2 < q_i$.
Input: $a(X) \in \mathcal{R}_q$.
Output: $d(X), c(X) \in \mathcal{R}_q$.

1: **for** $j \in [n]$ **do**
2: $\quad a'_j(X) := \left(\frac{q}{q_i}\ell_{i,j}(X)\right) a(X)$.
3: $\quad d'_j(X) := \mathsf{LUT}^{\mathsf{out}}_{\mathcal{L}_{\mathsf{mod},i,j}}(a'_j(X))$, where

$$\mathcal{L}_{\mathsf{mod},i,j} := \left((\frac{q}{q_i} k)\, \ell_{i,j}(X), [(k \mod B_p)\, \ell_{i,j}(X)]_{q_i} \mod q\mathcal{R} \right)_{k \in [0,(B_p-1)^2]}$$
$$\in (\mathcal{R}_q \times \mathcal{R}_q)^{(B_p-1)^2+1}.$$

4: $\quad c'_j(X) := \mathsf{LUT}^{\mathsf{out}}_{\mathcal{L}_{\mathsf{floor},i,j}}(a'_j(X))$, where

$$\mathcal{L}_{\mathsf{floor},i,j} := \left((\frac{q}{q_i} k)\, \ell_{i,j}(X), \left[\left\lfloor \frac{k}{B_p} \right\rfloor \ell_{i,j}(X)\right]_{q_i} \mod q\mathcal{R} \right)_{k \in [0,(B_p-1)^2]}$$
$$\in (\mathcal{R}_q \times \mathcal{R}_q)^{(B_p-1)^2+1}.$$

5: **end for**
6: $d(X) := \Sigma_{j\in[n]} d'_j(X)$ and $c(X) := \Sigma_{j\in[n]} c'_j(X)$.
7: **return** $(d(X), c(X))$.

1. *The below set is (a subset of) the admissible input set of* $C_{\mathsf{Mod\&Floor},i}$*:*

$$\mathcal{X}_{C_{\mathsf{Mod\&Floor},i},0} := \left\{ \mathit{NTT}_i^{-1}(\tilde{\mathbf{a}}) \mod q\mathcal{R} + q_i e_a(X) \right\},$$

where $\tilde{\mathbf{a}} \in [0, (B_p-1)^2]^n$ *and* $e_a(X) \in \mathcal{R}_q$.

2. *For any polynomial* $a(X) \in \mathcal{X}_{C_{\mathsf{Mod\&Floor},i},0}$*, there exist* $e_d(X), e_c(X) \in \mathcal{R}_q$ *such that*

$$d(X) = \mathit{NTT}_i^{-1}\left((\tilde{a}_j \mod B_p)_{j\in[n]}\right) \mod q\mathcal{R} + q_i e_d(X),$$
$$c(X) = \mathit{NTT}_i^{-1}\left(\left(\left\lfloor \frac{\tilde{a}_j}{B_p} \right\rfloor\right)_{j\in[n]}\right) \mod q\mathcal{R} + q_i e_c(X),$$
$$|e_d(X)|, |e_c(X)| \le n-1$$

hold, where

$$([\tilde{a}_j]_{q_i})_{j\in[n]} := \mathit{NTT}_i([a(X)]_{q_i}),$$
$$(d(X), c(X)) \leftarrow \mathit{Mod\&Floor}_{q,n,q_i,B_p}(a(X)).$$

3. *The non-free depth and addition width of* $C_{\mathsf{Mod\&Floor},i}$ *are* 2 *and* n*, respectively.*

Proof. ***Proof of the first property****: Recall that an input* $a(X) \in \mathcal{X}_{C_{\mathsf{Mod\&Floor},i},0}$ *is defined as below:*

$$a(X) := \mathsf{NTT}_i^{-1}([\tilde{\mathbf{a}}]_{q_i}) \mod q\mathcal{R} + q_i e_a(X),$$

where $\tilde{\mathbf{a}} := (\tilde{a}_1, \ldots, \tilde{a}_n)^T \in [0, (B_p-1)^2]^n$ *and* $e_a(X) \in \mathcal{R}_q$*. Since* $q_i \mid q$*, multiplying* $\frac{q}{q_i}$ *makes the* $q_i e_a(X)$ *term zero in* $q\mathcal{R}$*, i.e.,*

$$\frac{q}{q_i} a(X) \mod q\mathcal{R} = \frac{q}{q_i}\left(\mathsf{NTT}_i^{-1}\left([\tilde{\mathbf{a}}]_{q_i}\right)\right) \mod q\mathcal{R}.$$

By the orthogonality and idempotence of $[\ell_{i,j}(X)]_{q_i} = \mathsf{NTT}_i^{-1}(\left[\mathbf{u}_{j,n}\right]_{q_i})$*, we have*

$$\begin{aligned} a'_j(X) &:= (\frac{q}{q_i} a(X))\ell_{i,j}(X) \mod q\mathcal{R} \\ &= \left(\frac{q}{q_i}\left[\tilde{a}_j\right]_{q_i}\right)\ell_{i,j}(X) \mod q\mathcal{R}. \end{aligned}$$

Since $\tilde{a}_j \in [0, (B_p-1)^2]$ *holds by the definition of* $\mathcal{X}_{C_{\mathsf{Mod\&Floor},i},0}$*, the key* $a'_j(X) \in \mathcal{R}_q$ *must exist both in* $\mathcal{L}_{\mathsf{mod},i,j}$ *and* $\mathcal{L}_{\mathsf{floor},i,j}$*. Therefore,* $C_{\mathsf{Mod\&Floor},i}(a(X)) \neq \perp$ *holds for any* $a(X) \in \mathcal{X}_{C_{\mathsf{Mod\&Floor},i},0}$*.*

Proof of the second property*: By the definitions of* $\mathcal{L}_{\mathsf{mod},i,j}$ *and* $\mathcal{L}_{\mathsf{floor},i,j}$*, it holds that*

$$\begin{aligned} [d(X)]_{q_i} &= \mathsf{NTT}_i^{-1}\left((\tilde{a}_j \mod B_p)_{j\in[n]}\right) \mod q_i\mathcal{R}, \\ [c(X)]_{q_i} &= \mathsf{NTT}_i^{-1}\left(\left(\left\lfloor \frac{\tilde{a}_j}{B_p} \right\rfloor\right)_{j\in[n]}\right) \mod q_i\mathcal{R}. \end{aligned}$$

However, since the outputs $d(X)$ *and* $c(X)$ *are computed in* $\mathcal{R}_q$*, we have*

$$\begin{aligned} d(X) &= \mathsf{NTT}_i^{-1}\left((\tilde{a}_j \mod B_p)_{j\in[n]}\right) \mod q\mathcal{R} + q_i e_d(X), \\ c(X) &= \mathsf{NTT}_i^{-1}\left(\left(\left\lfloor \frac{\tilde{a}_j}{B_p} \right\rfloor\right)_{j\in[n]}\right) \mod q\mathcal{R} + q_i e_c(X) \end{aligned}$$

for some $e_d(X)$ *and* $e_c(X)$*. It also holds that*

$$|e_d(X)|, |e_c(X)| \leq \left\lfloor \frac{n(q_i-1)}{q_i} \right\rfloor = n-1,$$

where the inequality follows since $d(X)$ *and* $c(X)$ *are sum of* n *polynomials in* $q_i\mathcal{R}$*, and the equation holds since* $n < q_i$ *by the choice of* n *and* q_i*.*

Proof of the third property*: The loop of* j *in the* $\mathsf{Mod\&Floor}$ *algorithm contains scalar multiplications in the first non-free layer and the LUT evaluations in the second non-linear layer. Since there is no dependence between processes for* j *and* j' *such that* $j \neq j'$*, the non-free depth is 2. Each circuit output is a sum of* n *outputs of the LUT evaluations, and all non-free gate takes only one input wire. Hence, the addition width is* n*.*

Algorithm 4 $\mathsf{MultiPre}_{q,n,B_p}$

Parameter Settings: $q = \prod_{i\in[h]} q_i$, $w_p = \max_{i\in[h]} \lceil \log_{B_p} q_i \rceil$, and $q_i^\star \in \mathbb{Z}, \tilde{q}_i \in \mathbb{Z}_{q_i}$ for each $i \in [h]$ defined in Subsection 3.4.
Input: $\tilde{\mathbf{a}} \in \mathbb{Z}_q^n$.
Output: $\mathbf{a} \in \mathcal{R}_q^{w_p}$.

1: **for** $j \in [n]$ **do**
2: $\quad ([\tilde{a}_j]_{q_1}, \ldots, [\tilde{a}_j]_{q_h})^T := \mathsf{CRT}(\tilde{a}_j)$.
3: $\quad$ **for** $i \in [h]$ **do**
4: $\quad\quad ([\tilde{a}_{1,j}]_{q_i}, \ldots, [\tilde{a}_{w_p,j}]_{q_i})^T := \mathsf{digits}_{B_p,w_p}([\tilde{a}_j]_{q_i})$.
5: $\quad$ **end for**
6: **end for**
7: $\mathbf{A} := \begin{pmatrix} \sum_{i\in[h]} (q_i^\star \tilde{q}_i) \mathsf{NTT}_i^{-1}(([a_{1,j}]_{q_i})_{j\in[n]}^T) \mod q\mathcal{R} \\ \vdots \\ \sum_{i\in[h]} (q_i^\star \tilde{q}_i) \mathsf{NTT}_i^{-1}(([a_{w_p,j}]_{q_i})_{j\in[n]}^T) \mod q\mathcal{R} \end{pmatrix}$.
8: **return** $\mathbf{A}$.

Definition 4 (Multi-precision representation). *Let $q, n \in \mathbb{N}$ be the modulus and the degree of the ring $\mathcal{R}_q$ compatible with the double-CRT representation as described in Subsection 3.4. We write $q = \prod_{i\in[h]} q_i$ and assume that $\min_{i\in[h]} q_i > n$ holds. Fix a positive integer $B_p \geq 2$ that satisfies $(B_p - 1)^2 < \min_{i\in[h]} q_i$. Set $w_p = \max_{i\in[h]} \lceil \log_{B_p} q_i \rceil$. A multi-precision representation of a vector $\tilde{\mathbf{a}} \in \mathbb{Z}_q^n$ is defined by the output $\mathbf{a} \leftarrow \mathsf{MultiPre}(\tilde{\mathbf{a}}) \in \mathcal{R}_q^{w_p}$ (Algorithm 4). For every $l \in [w_p]$ and $i \in [h]$, the i-th CRT slot of the l-th polynomial of $\mathbf{a}$, namely $[a_l(X)]_{q_i}$, encodes the l-th digits of $[\tilde{\mathbf{a}}]_{q_i} \in \mathbb{Z}_{q_i}^n$ in the evaluation slots.*

Lemma 7. *We keep the same parameter settings in Definition 4. For any circuit $C_Z \in \mathcal{C}_{\mathbb{Z}_q^n,L,M}$ (Definition 3), there exists a corresponding circuit $C_R \in \mathcal{C}_{\mathcal{R}_q,w_p+,M}$ such that the following properties hold:*

- *The below set is (a subset of) the admissible input sets of C_R:*

$$\mathcal{X}_{C_R,B_{\times,R}} := \left\{ \begin{pmatrix} 1 \\ \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_L \end{pmatrix} \right\},$$

  where $\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_L \in \mathbb{Z}_q^n$, $\mathbf{x}_\ell \leftarrow \mathsf{MultiPre}_{q,n,B_p}(\tilde{\mathbf{x}}_\ell)$ for every $\ell \in [L]$, and $B_{\times,R} := \max_{i\in[h]} B_{\times,q_i}$.
- *For any L vectors $\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_L \in \mathbb{Z}_q^n$, it holds that*

$$\begin{pmatrix} \mathsf{NTT}_{1,\ldots,h}^{-1}(\tilde{\mathbf{y}}_1) \\ \vdots \\ \mathsf{NTT}_{1,\ldots,h}^{-1}(\tilde{\mathbf{y}}_M) \end{pmatrix} = C_R \begin{pmatrix} 1 \\ \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_L \end{pmatrix},$$

where $(\tilde{\mathbf{y}}_\mu)_{\mu\in[M]}^T := C_Z((\tilde{\mathbf{x}}_\ell)_{\ell\in[L]}^T)$ *and* $\mathbf{x}_\ell \leftarrow \mathsf{MultiPre}_{q,n,B_p}(\tilde{\mathbf{x}}_\ell)$ *for every* $\ell \in [L]$*. The above inputs to* C_R *obviously belong to* $\mathcal{X}_{C_R,B_{\times,R}}$*.*

– *The depth of* C_R *is bounded by [TODO]*

Proof. We first present how the norm-bounded circuit $C_R \in \mathcal{C}_{\mathcal{R}_q,w_pL,M}$ *is defined for the given* $C_Z \in \mathcal{C}_{\mathbb{Z}_q^n,L,M}$*. The inputs to* C_R *are processed sequentially in the following three stages:*

Stage 1: Input decomposition. *Recall that* C_R *takes as input* w_pL *polynomials* $(x_{\ell,1}(X),\ldots,x_{\ell,w_p}(X))_{\ell\in[L]}^T \in \mathcal{R}_q^{w_pL}$*. In the first layer, for every* $i \in [h], \ell \in [L], l \in [w_p]$*,* C_R *computes* $x_{i,\ell,l}(X), _ \leftarrow \mathsf{Mod\&Floor}_{q,n,q_i,B_p}(x_{\ell,l}(X))$*. Since the inputs are multi-precision of each input vector* $\tilde{\mathbf{x}}_\ell := (\tilde{x}_{\ell,j})_{j\in[n]} \in \mathbb{Z}_q^n$*, as described in Definition 4,* $\mathsf{NTT}_i([x_{\ell,l}(X)]_{q_i})$ *is equivalent to the* l*-th digits of* $[\tilde{\mathbf{x}}_\ell]_{q_i}$*. Hence, it follows that*

$$\mathsf{NTT}_i([x_{\ell,l}(X)]_{q_i}) = ([\tilde{x}_{\ell,l,j}]_{q_i})_{j\in[n]} \in [0, B_p-1]^n,$$

where $([\tilde{x}_{\ell,l,j}]_{q_i})_{l\in[w_p]} := \mathsf{digits}_{B_p,w_p}([\tilde{x}_{\ell,j}]_{q_i})$ *for every* $j \in [n]$*. By Lemma 6, it holds that* $x_{\ell,l}(X) \in \mathcal{X}_{C_{\mathsf{Mod\&Floor},i},0}$*, and we obatin*

$$\begin{aligned} x_{\ell,l,i}(X) &= \mathsf{NTT}_i^{-1}\left(([\tilde{x}_{\ell,l,j}]_{q_i} \mod B_p)_{j\in[n]}\right) \mod q\mathcal{R} + q_i e_{x_{\ell,l,i}}(X) \\ &= \mathsf{NTT}_i^{-1}\left(([\tilde{x}_{\ell,l,j}]_{q_i})_{j\in[n]}\right) \mod q\mathcal{R} + q_i e_{x_{\ell,l,i}}(X), \end{aligned}$$

where $|e_{x_{\ell,l,i}}(X)| \le n-1$*, and the last equation holds because* $[\tilde{x}_{\ell,l,j}]_{q_i} \mod B_p = [\tilde{x}_{\ell,l,j}]_{q_i}$*. Consequently, for every* $\ell \in [L]$*, the first stage outputs a matrix* $\mathbf{X}_\ell := \begin{pmatrix} x_{\ell,1,1}(X) & \ldots & x_{\ell,1,h}(X) \\ \vdots & \ddots & \vdots \\ x_{\ell,w_p,1}(X) & \ldots & x_{\ell,w_p,h}(X) \end{pmatrix} \in \mathcal{R}_q^{w_p\times h}$*.*

Stage 2: Full-domain arithmetic operation. *In the second stage, we traverse* C_Z *from its first layer to its last, executing the operation corresponding to each gate in* C_Z *by composing the norm-bounded circuit gate types specified in Definition 2. Let* $\tilde{\mathbf{z}}_w := (\tilde{z}_{w,j})_{j\in[n]} \in \mathbb{Z}_q^n$ *be a vector carried by wire* w *in* C_Z *during the execution of* $C_Z((\tilde{\mathbf{x}}_\ell)_{\ell\in[L]}^T)$*. For any wire* w *in* C_Z*, we say that a matrix* $\mathbf{Z}_w \in \mathcal{R}_q^{w_p\times h}$ *corresponds to wire* w *if the following holds:*

$$\mathbf{Z}_w = \begin{pmatrix} \left(\mathsf{NTT}_i^{-1}\left(([\tilde{z}_{w,1,j}]_{q_i})_{j\in[n]}\right) \mod q\mathcal{R} + q_i e_{z_{w,1,i}}(X)\right)_{i\in[h]}^T \\ \vdots \\ \left(\mathsf{NTT}_i^{-1}\left(([\tilde{z}_{w,w_p,j}]_{q_i})_{j\in[n]}\right) \mod q\mathcal{R} + q_i e_{z_{w,w_p,i}}(X)\right)_{i\in[h]}^T \end{pmatrix}, \quad (8)$$

where $([\tilde{z}_{w,l,j}]_{q_i})_{l\in[w_p]} := \mathsf{digits}_{B_p,w_p}([\tilde{z}_{w,j}]_{q_i})$ *for every* $i \in [h]$ *and* $j \in [n]$*.*

Any input wire u *of a gate in the first layer of* C_Z *is, for some* ℓ*, also the* ℓ*-th input wire of* C_Z*. Accordingly, the matrix* $\mathbf{Z}_u$ *corresponding to wire* u *is defined by the matrix* $\mathbf{X}_\ell$ *output from the first stage.*

For any gate g_w in C_Z with input wires u and v and output wire w, assume there exist matrices $\mathbf{Z}_u, \mathbf{Z}_v \in \mathcal{R}_q^{w_p \times h}$ corresponding to wires u and v, respectively. Then there is also a matrix $\mathbf{Z}_w \in \mathcal{R}_q^{w_p \times h}$ corresponding to wire w as follows:

- *If g_w represents $\tilde{\mathbf{z}}_u \boxplus \tilde{\mathbf{z}}_v$,*
- *If g_w represents $\tilde{\mathbf{z}}_u \boxminus \tilde{\mathbf{z}}_v$,*
- *If g_w represents $\tilde{\mathbf{z}}_u \boxtimes \tilde{\mathbf{z}}_v$,*

Stage 3: Output composition. *From the second stage, we can obtain a matrix $\mathbf{Y}_\mu$ corresponding to the μ-th output wire of C_Z for each $\mu \in [M]$, which is one of gate outputs in the last layer.*

□

Theorem 2. *Let $q, n \in \mathbb{N}$ be the modulus and the degree of the ring $\mathcal{R}_q$ compatible with the double-CRT representation as described in Subsection 3.4. We write $q = \prod_{i\in[h]} q_i$ and assume that $\min_{i\in[h]} q_i > n$ holds. Fix a positive integer $B_g, B_p \geq 2$ that satisfies $(B_p - 1)^2 < \min_{i\in[h]} q_i$. Set $m_g := \left\lceil \log_{B_g} q \right\rceil$, $m_b := \Theta(\log_2 q)$, $\sigma_b := \mathcal{O}(n \log_2 q)\, \omega(\sqrt{\log_2 m_b})$, and $w_p = \max_{i\in[h]} \lceil \log_{B_p} q_i \rceil$. For a class $\mathcal{C}_{\mathbb{Z}_q^n, L, M}$ (Definition 3), there exist two deterministic algorithms that satisfy key-homomorphic properties defined as below.*

- $\mathsf{EvalC}_{\mathbb{Z}_q^n}(\mathbf{A}_{\mathsf{inp}}, \mathbf{b}_{\sigma_b}^{-1}, C_Z) \to ((\mathbf{a}_{\mathsf{out},i})_{i\in[M]}, \mathcal{K}_{C_Z,\mathsf{LUT}})$*: This takes as input a matrix $\mathbf{A}_{\mathsf{inp}} \in \mathcal{R}_q^{m_g \times (1+w_p L)}$, a trapdoor $\mathbf{b}_{\sigma_b}^{-1}$, and a circuit $C_Z \in \mathcal{C}_{\mathbb{Z}_q^n, L, M}$. This outputs a matrix $(\mathbf{a}_{\mathsf{out},i})_{i\in[M]} \in \mathcal{R}_q^{m_g \times M}$ and a set of matrices $\mathcal{K}_{C_Z,\mathsf{LUT}} \subset \bigcup_{T\in[(B_p-1)^2+1]} \mathcal{R}_q^{(m_b+m_g)T \times m_g}$.*
- $\mathsf{EvalCX}_{\mathbb{Z}_q^n}(\mathbf{A}_{\mathsf{inp}}, C_Z, \mathcal{K}_{C,\mathsf{LUT}}, \tilde{\mathbf{X}}_L) \to \mathbf{H}_{C_Z,x}$*: This takes as input a matrix $\mathbf{A}_{\mathsf{inp}} \in \mathcal{R}_q^{m_g \times (1+w_p L)}$, a circuit $C_Z \in \mathcal{C}_{\mathbb{Z}_q^n, L, M}$, a set of matrices $\mathcal{K}_{C,\mathsf{LUT}} \subset \bigcup_{T\in[(B_p-1)^2+1]} \mathcal{R}_q^{(m_b+m_g)T \times m_g}$, and a matrix $\tilde{\mathbf{X}}_L \in \mathbb{Z}_q^{n \times L}$. This outputs a matrix $\mathbf{H}_{C,x} \in \mathcal{R}_q^{(m_b+m_g(1+w_p L)) \times m_g M}$.*

For any matrix $\mathbf{A}_{\mathsf{inp}} \in \mathcal{R}_q^{m_g \times (1+w_p L)}$, any pair of a circuit $C_Z \in \mathcal{C}_{\mathbb{Z}_q^n, L, M}$ and a matrix $\tilde{\mathbf{X}}_L := (\tilde{\mathbf{x}}_i)_{i\in[L]}^T \in \mathbb{Z}_q^{n \times L}$,

$$\begin{aligned} &\left(\mathbf{b}^T, \mathsf{flatten}\left(\mathbf{A}_{\mathsf{inp}}\right)^T - (1, \mathbf{x}_{w_p L}^T) \otimes \mathbf{g}^T\right) \mathbf{H}_{C_Z,x,i} \\ &= \left(\mathbf{a}_{\mathsf{out},1}^T - y_1(X)\mathbf{g}^T, \ldots, \mathbf{a}_{\mathsf{out},M}^T - y_M(X)\mathbf{g}^T\right), \end{aligned} \tag{9}$$

holds, where

$$
\begin{aligned}
&(\mathbf{b}, \mathbf{b}_{\sigma_b}^{-1}) \leftarrow \mathsf{TrapGen}(1^n, 1^{m_b}, 1^g),\\
&((\mathbf{a}_{\mathsf{out},\mu})_{\mu\in[M]}, \mathcal{K}_{C,\mathsf{LUT}}) \leftarrow \mathsf{EvalC}_{\mathbb{Z}_q^n}(\mathbf{A}_{\mathsf{inp}}, \mathbf{b}_{\sigma_b}^{-1}, C_Z),\\
&(\mathbf{H}_{C,x,\mu})_{\mu\in[M]} \leftarrow \mathsf{EvalCX}_{\mathbb{Z}_q^n}(\mathbf{A}_{\mathsf{inp}}, C_Z, \mathcal{K}_{C,\mathsf{LUT}}, \tilde{\mathbf{X}}_L),\\
&(\tilde{\mathbf{y}}_\mu)_{\mu\in[M]} := C_Z((\tilde{\mathbf{x}}_\ell)_{\ell\in[L]}),\\
&\forall \ell \in [L] \quad \mathbf{x}_\ell \leftarrow \mathsf{MultiPre}_{q,n,B_p}(\tilde{\mathbf{x}}_\ell),\\
&\mathbf{x}_{w_pL}^T := \left(\mathbf{x}_1^T, \ldots, \mathbf{x}_L^T\right) \in \mathcal{R}_q^{1\times w_pL},\\
&\begin{pmatrix} y_1(X) \\ \vdots \\ y_M(X) \end{pmatrix} := \begin{pmatrix} \mathsf{NTT}_{1,\ldots,h}^{-1}(\tilde{\mathbf{y}}_1) \\ \vdots \\ \mathsf{NTT}_{1,\ldots,h}^{-1}(\tilde{\mathbf{y}}_M) \end{pmatrix}.
\end{aligned}
$$

Additionally, it holds that

$$\left\|\mathbf{H}_{C,X}\right\|_\infty \leq [TODO] \tag{10}$$

6 KP-ABE for Full-domain Arithmetic Circuits

6.1 Definition

We define KP-ABE for the class $\mathcal{C}_{\mathbb{Z}_q^n,L,1}$ (Definition 3) and message space $\mathcal{M}$ with the following syntax:

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{mpk}, \mathsf{msk})$: This takes as input a security parameter λ and outputs a master public key mpk and a master secret key msk.
- $\mathsf{Enc}(\mathsf{mpk}, \tilde{\mathbf{X}}, \mu) \to \mathsf{ct}$: This takes as input a master public key mpk, an attribute $\tilde{\mathbf{X}} \in \mathbb{Z}_q^{n\times L}$, a message $\mu \in \mathcal{M}$. This outputs a ciphertext ct.
- $\mathsf{KeyGen}(\mathsf{msk}, C) \to \mathsf{dk}_C$: This takes as input a master secret key msk and a circuit $C \in \mathcal{C}_{\mathbb{Z}_q^n,L,1}$. This outputs a decryption key dk_C.
- $\mathsf{Dec}(\mathsf{mpk}, \tilde{\mathbf{X}}, C, \mathsf{ct}, \mathsf{dk}_C) \to \mu$ or $\perp$: This takes as input a master public key mpk, an attribute $\tilde{\mathbf{X}} \in \mathbb{Z}_q^{n\times L}$, a circuit $C \in polyCFVABE$, a ciphertext ct, and a decryption key dk_C. This outputs a message $\mu \in \mathcal{M}$ or a symbol $\perp$.

Definition 5 (Correctness of KP-ABE). *An KP-ABE scheme* $\mathsf{KP\text{-}ABE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ *for a class of full-domain vector arithmetic circuits* $\mathcal{C}_{\mathbb{Z}_q^n,L,1}$ *and a messasge class* $\mathcal{M}$ *is said to be correct if for all* $\lambda \in \mathbb{N}$*, messages* $\mu \in \mathcal{M}$*, attributes* $\tilde{\mathbf{X}} \in \mathbb{Z}_q^{n\times L}$*, and circuits* $C \in \mathcal{C}_{\mathbb{Z}_q^n,L,1}$ *such that* $C(\tilde{\mathbf{X}}) = \mathbf{0}_n$*, it holds that*

$$\Pr\left[\mathsf{Dec}(\mathsf{mpk}, \tilde{\mathbf{X}}, C, \mathsf{ct}, \mathsf{dk}_C) = \mu \;:\; \begin{array}{l}(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda),\\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \tilde{\mathbf{X}}, \mu),\\ \mathsf{dk}_C \leftarrow \mathsf{KeyGen}(\mathsf{msk}, C)\end{array}\right] = 1 - \mathsf{negl}(\lambda).$$

We adopt the definition of the selective security for the KP-ABE scheme from [7] with necessary modifications.

Definition 6 (Selective Security for KP-ABE). *An KP-ABE scheme* $\mathsf{KP\text{-}ABE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ *for a class of full-domain vector arithmetic circuits* $\mathcal{C}_{\mathbb{Z}_q^n, L, 1}$ *and a messasge class* $\mathcal{M}$ *is said to be selectively secure if for all PPT adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$*, there is a negligible function* $\mathsf{negl}(\lambda)$ *such that*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{sel}}_{\mathsf{KP\text{-}ABE},\mathcal{A}}(\lambda) &:= \left|\Pr\left[\mathsf{EXP}^{0}_{\mathsf{KP\text{-}ABE},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{EXP}^{1}_{\mathsf{KP\text{-}ABE},\mathcal{A}}(\lambda) = 1\right]\right| \\ &\leq \mathsf{negl}(\lambda),\end{aligned}$$

where for each $b \in \{0,1\}$ *and* $\lambda \in \mathbb{N}$*. The experiment* $\mathsf{EXP}^{b}_{\mathsf{KP\text{-}ABE},\mathcal{A}}(\lambda)$ *is defined as follows:*

1. $(\tilde{\mathbf{X}}^{\star}, \mathsf{state}_1) \leftarrow \mathcal{A}_1(\lambda)$*, where* $\tilde{\mathbf{X}}^{\star} \in \mathbb{Z}_q^{n \times L}$
2. $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^{\lambda})$
3. $(\mu_0, \mu_1, \mathsf{state}_2) \leftarrow \mathcal{A}_2^{\mathsf{KG}(\mathsf{msk}, \tilde{\mathbf{X}}^{\star}, \cdot)}(\mathsf{mpk}, \mathsf{state}_1)$
4. $\mathsf{ct}^{*} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \tilde{\mathbf{X}}^{\star}, \mu_b)$
5. $b' \leftarrow \mathcal{A}_3^{\mathsf{KG}(\mathsf{msk}, \tilde{\mathbf{X}}^{\star}, \cdot)}(\mathsf{ct}^{*}, \mathsf{state}_2)$
6. *Output* b'*,*

where $\mathsf{KG}(\mathsf{msk}, \tilde{\mathbf{X}}^{\star}, C)$ *on input a circuit* $C \in \mathcal{C}_{\mathbb{Z}_q^n, L, 1}$ *returns a decryption key* $\mathsf{dk}_C \leftarrow \mathsf{KeyGen}(\mathsf{msk}, C)$ *if* $C(\tilde{\mathbf{X}}^{\star}) \neq \mathbf{0}_n$ *and* $\perp$ *otherwise.*

6.2 Construction

Using the key-homomorphic algorithms introduced in Theorem 2, we construct a KP-ABE scheme for the class of full-domain vector arithmetic circuits with message class $\mathcal{M} = \mathcal{R}_3$; see Algorithms 5, 6, 7, and 8. These algorithms employ the following parameters in common:

Parameters:

- The security parameter $\lambda \in \mathbb{N}$.
- The modulus $q = \prod_{i \in [h]} q_i$ and the power-of-two degree n such that they are compatible with the double-CRT representation as described in Subsection 3.4, and $\min_{i \in [h]} q_i > n$ and Inequality 1 hold.
- Positive integers $B_g, B_p \geq 2$ that satisfies $(B_p - 1)^2 < \min_{i \in [h]} q_i$.
- Positive integers $m_g := \left\lceil \log_{B_g} q \right\rceil$, $w_p = \max_{i \in [h]} \lceil \log_{B_p} q_i \rceil$, and $m_b := \Theta(\log_2 q)$ such that m_b satisfies Inequality 1.
- Input size $L \in \mathbb{N}$.
- Gaussian parameters $\sigma_b := \mathcal{O}(n \log_2 q)\, \omega(\sqrt{\log_2 m_b})$ and σ_e.

Lemma 8. *The KP-ABE scheme* $\mathsf{KP\text{-}ABE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ *described in Algorithms 5, 6, 7, and 8 is correct according to Definition 5.*

Algorithm 5 ABE.Setup

Parameter Settings: Parameters defined in Subsection 6.2.
Input: 1^λ.
Output: $\mathsf{mpk}, \mathsf{msk}$.

$\mathbf{A}_{\mathsf{inp}} \leftarrow\!\!\$\ \mathcal{R}_q^{m_g \times (1+w_p L)}$.
$(\mathbf{b}, \mathbf{b}_\sigma^{-1}) \leftarrow \mathsf{TrapGen}(1^n, 1^{m_b}, 1^q)$.
$u(X) \leftarrow \mathcal{R}_q$.
$\mathsf{mpk} := (\mathbf{A}_{\mathsf{inp}}, \mathbf{b}, u(X))$.
$\mathsf{msk} := (\mathbf{A}_{\mathsf{inp}}, \mathbf{b}, u(X), \mathbf{b}_\sigma^{-1})$.
return mpk, msk.

Algorithm 6 ABE.Enc

Parameter Settings: Parameters defined in Subsection 6.2.
Input: $\mathsf{mpk}, \tilde{\mathbf{X}} \in \mathbb{Z}_q^{n \times L}, \mu \in \mathcal{R}_2$.
Output: ct.

Parse mpk as $(\mathbf{A}_{\mathsf{inp}}, \mathbf{b}, u(X))$.
Parse $\tilde{\mathbf{X}}$ as $(\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_L)$, where $\tilde{\mathbf{x}}_i \in \mathbb{Z}_q^n$ for every $i \in [L]$.
For every $i \in [L]$, $\mathbf{x}_i \leftarrow \mathsf{MultiPre}_{q,n,B_p}(\tilde{\mathbf{x}}_i)$.
$\mathbf{x}_{w_p L}^T := \left(\mathbf{x}_1^T, \ldots, \mathbf{x}_L^T\right) \in \mathcal{R}_q^{1 \times w_p L}$.
$s(X) \leftarrow\!\!\$\ \mathcal{R}_3$ and $\mathbf{e}_b \leftarrow\!\!\$\ \mathcal{D}_{\mathcal{R},\sigma_e,\leq\sigma_e\sqrt{\lambda}}^{m_b}$.
$\mathbf{R} \leftarrow\!\!\$\ \mathcal{R}_3^{m_b \times (1+w_p L) m_g}$.
$\mathbf{c}_b^T := s(X)\mathbf{b}^T + \mathbf{e}_b^T$.
$\mathbf{c}_a^T := s(X)\left(\mathsf{flatten}(\mathbf{A}_{\mathsf{inp}})^T - (1, \mathbf{x}_{w_p L}^T) \otimes \mathbf{g}^T\right) + \mathbf{e}_b^T \mathbf{R}$.
$e_u(X) \leftarrow\!\!\$\ \mathcal{D}_{\mathcal{R},\sigma_e,\leq\sigma_e\sqrt{\lambda}}$ and $c_u(X) := s(X)u(X) + e_u(X) + \lceil \frac{q}{2} \rfloor \mu$.
$\mathsf{ct} := (\mathbf{c}_b, \mathbf{c}_a, c_u(X))$.
return ct.

Proof. *We show that* $\mathsf{Dec}(\mathsf{mpk}, \tilde{\mathbf{X}}, C, \mathsf{ct}, \mathsf{dk}_C) = \mu$ *holds if* $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \tilde{\mathbf{X}}, \mu)$, $\mathsf{dk}_C \leftarrow \mathsf{KeyGen}(\mathsf{msk}, C)$, *and* $C(\tilde{\mathbf{X}}) = \mathbf{0}_n$ *hold. We have*

$$\begin{aligned}
\mathbf{c}_y^T &= (\mathbf{c}_b^T, \mathbf{c}_a^T)\mathbf{H}_{C,x} \\
&= s(X)\left(\mathbf{b}^T, \mathit{flatten}(\mathbf{A}_{\mathit{inp}})^T - (1, \mathbf{x}_{w_p L}^T) \otimes \mathbf{g}^T\right)\mathbf{H}_{C,x} + \mathbf{e}_b^T\left(\mathbf{I}_{m_b}, \mathbf{R}\right)\mathbf{H}_{C,x} \\
&= s(X)\left(\mathbf{a}_{\mathit{out}}^T - y(X)\mathbf{g}^T\right) + \mathbf{e}_b^T\left(\mathbf{I}_{m_b}, \mathbf{R}\right)\mathbf{H}_{C,x} \\
&= s(X)\mathbf{a}_{\mathit{out}}^T + \mathbf{e}_y^T,
\end{aligned}$$

where

$$\begin{aligned}
&\tilde{\mathbf{y}} := C(\tilde{\mathbf{X}}), \quad \mathbf{e}_y^T := \mathbf{e}_b^T\left(\mathbf{I}_{m_b}, \mathbf{R}\right)\mathbf{H}_{C,x}, \\
&y(X) := \mathit{NTT}_{1,\ldots,h}^{-1}(\tilde{\mathbf{y}}).
\end{aligned}$$

In the above derivation, the second equality follows from Lemma 2, and the final equality holds since $C(\tilde{\mathbf{X}}) = \mathbf{0}_n$.

Algorithm 7 ABE.KeyGen

Parameter Settings: Parameters defined in Subsection 6.2.
Input: $\mathsf{msk}, C \in \mathcal{C}_{\mathbb{Z}_q^n, L, 1}$.
Output: dk_C.

1: Parse msk as $(\mathbf{A}_{\mathsf{inp}}, \mathbf{b}, u(X), \mathbf{b}_\sigma^{-1})$.
2: $(\mathbf{a}_{\mathsf{out}}, \mathcal{K}_{C,\mathsf{LUT}}) \leftarrow \mathsf{EvalC}_{\mathbb{Z}_q^n}(\mathbf{A}_{\mathsf{inp}}, \mathbf{b}_{\sigma_b}^{-1}, C)$.
3: $\mathbf{k}_{C,\,\mathsf{low}} \leftarrow\!\$\ \mathcal{D}^{m_g}_{\mathcal{R},\sigma_b,\leq\sigma_b\sqrt{\lambda}}$.
4: $t(X) := u(X) - \mathbf{a}_{\mathsf{out}}^T \mathbf{k}_{C,\,\mathsf{low}}$.
5: $\mathbf{k}_{C,\,\mathsf{high}} \leftarrow\!\$\ \mathbf{b}_\sigma^{-1}(t(X))$.
6: $\mathbf{k}_C = \begin{pmatrix} \mathbf{k}_{C,\,\mathsf{high}} \\ \mathbf{k}_{C,\,\mathsf{low}} \end{pmatrix}$.
7: $\mathsf{dk}_C = (\mathcal{K}_{C,\mathsf{LUT}}, \mathbf{k}_C)$.
8: **return** dk_C.

Algorithm 8 ABE.Dec

Parameter Settings: Parameters defined in Subsection 6.2.
Input: $\mathsf{mpk}, \tilde{\mathbf{X}} \in \mathbb{Z}_q^{n\times L}, C \in \mathcal{C}_{\mathbb{Z}_q^n, L, 1}, \mathsf{ct}, \mathsf{dk}_C$.
Output: $\mu \in \mathcal{R}_2$ or $\perp$.

1: Parse mpk as $(\mathbf{A}_{\mathsf{inp}}, \mathbf{b}, u(X))$.
2: Parse ct as $(\mathbf{c}_b, \mathbf{c}_a, c_u(X))$.
3: Parse dk_C as $(\mathcal{K}_{C,\mathsf{LUT}}, \mathbf{k}_C)$,
4: $\mathbf{H}_{C,x} \leftarrow \mathsf{EvalC}_{\mathbb{Z}_q^n}(\mathbf{A}_{\mathsf{inp}}, C, \mathcal{K}_{C,\mathsf{LUT}}, \tilde{\mathbf{X}})$.
5: $\mathbf{c}_y^T := (\mathbf{c}_b^T, \mathbf{c}_a^T)\mathbf{H}_{C,x}$.
6: $z(X) := c_u(X) - (\mathbf{c}_b^T, \mathbf{c}_y^T)\mathbf{k}_C$.
7: $\mu := \mathsf{Round}_{q/4}(z(X))$.
8: **return** μ.

By the property of the preimage, it follows that

$$\begin{aligned}(\mathbf{c}_b^T, \mathbf{c}_y^T)\mathbf{k}_C =& \mathbf{c}_b^T \mathbf{k}_{C,\,\mathit{high}} + \mathbf{c}_y^T \mathbf{k}_{C,\,\mathit{low}} \\ =& s(X)\left(u(X) - \mathbf{a}_{\mathit{out}}^T \mathbf{k}_{C,\,\mathit{low}}\right) + \mathbf{e}_b^T \mathbf{k}_{C,\,\mathit{high}} \\ &+ s(X)\mathbf{a}_{\mathit{out},i}^T \mathbf{k}_{C,\,\mathit{low}} + \mathbf{e}_y^T \mathbf{k}_{C,\,\mathit{low}} \\ =& s(X)u(X) + \mathbf{e}_b^T \mathbf{k}_{C,\,\mathit{high}} + \mathbf{e}_y^T \mathbf{k}_{C,\,\mathit{low}}.\end{aligned}$$

Therefore, we can obtain

$$\begin{aligned}z(X) =& c_u(X) - (\mathbf{c}_b^T, \mathbf{c}_y^T)\mathbf{k}_C \\ =& s(X)u(X) + e_u(X) + \left\lceil \frac{q}{2} \right\rfloor \mu - \left(s(X)u(X) + \mathbf{e}_b^T \mathbf{k}_{C,\,\mathit{high}} + \mathbf{e}_y^T \mathbf{k}_{C,\,\mathit{low}}\right) \\ =& \left\lceil \frac{q}{2} \right\rfloor \mu + e_z(X),\end{aligned}$$

where $e_z(X) := e_u(X) - \mathbf{e}_b^T \mathbf{k}_{C,\,\mathit{high}} - \mathbf{e}_y^T \mathbf{k}_{C,\,\mathit{low}}$.

Recall that, for every $i \in [n]$*, the* i*-th coefficient of* $\mathsf{Round}_{q/4}(z(X))$ *is 1 if that of* $z(X)$ *is in* $[\frac{q}{4}, \frac{3q}{4})$ *and 0 otherwise. Since* $\mu \in \mathcal{R}_2$*, it suffices to confirm that* $|e_z(X)| < \frac{q}{4}$*. [TODO]* □

Theorem 3. *Assuming that the ring-LWE assumption (Assumption 1) is hard with integers q, n, the secret distribution $\mathcal{R}_3$, and the error distribution $\mathcal{D}_{\mathcal{R},\sigma_e,\leq\sigma_e\sqrt{\lambda}}$, the KP-ABE scheme $\mathsf{KP\text{-}ABE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ described in Algorithms 5, 6, 7, and 8 is selectively secure according to Definition 6.*

Proof. ***Proof sketch****: We follow the proof of Theorem4.2 in [7], except that we must also show that the preimages produced by $\mathsf{EvalLut}$ (Algorithm 1) can be simulated without access to the trapdoor $\mathbf{b}_\sigma^{-1}$. For $k = \mathsf{LUT}_{\mathcal{L}}^{\mathsf{idx}}(z(X))$, we will show that the target vector of the preimage $\mathbf{K}_{w,k,\,\mathsf{high}}$, defined by $\mathbf{p}_{w,k}$ in Algorithm 1, is indistinguishable from a uniformly random vector; therefore, the preimage $\mathbf{K}_{w,k}$ is statistically indistinguishable from a matrix $\mathbf{K}_{w,k} \leftarrow\!\$\ \mathcal{D}_{\mathcal{R},\sigma_b,\leq\sigma_b\sqrt{\lambda}}^{(m_b+m_g)\times m_g}$ from Lemma 2. For $k \neq \mathsf{LUT}_{\mathcal{L}}^{\mathsf{idx}}(z(X))$, we will invoke $\mathsf{SampLeft}$ to publicly sample the preimage $\mathbf{K}_{w,k,\,\mathsf{high}}$ without the trapdoor.*

We define a sequence of hybrids as follows.

- ***Hyb 0****: This hybrid corresponds to the real distribution in $\mathsf{EXP}^{\,b}_{\mathsf{KP\text{-}ABE},\mathcal{A}}(\lambda)$.*
- ***Hyb 1****: For each query $C \in \mathcal{C}_{\mathcal{R}_q,L,M}$ to KG, let $C_{\mathcal{R}_q} \in \mathcal{C}_{\mathcal{R}_q,Lw_p,1}$ denote a norm-bounded polynomial circuit corresponding to C according to Lemma 7. For every output wire w of the LUT evaluation gate g_w in $C_{\mathcal{R}_q}$, which is associated to a LUT $\mathcal{L}_w \in (\mathcal{R}_q \times \mathcal{R}_q)^{R_w}$, this hybrid replaces a vector $\mathbf{a}_w \in \mathcal{R}_q^{m_g}$ output by $\mathsf{EvalLut}(\mathbf{a}_u, \mathbf{b}_{\sigma_b}^{-1}, \mathcal{L}_w)$ as follows:*

$$\mathbf{a}_w^T := \bar{\mathbf{a}}_w^T + \bar{y}_{k^\star}(X)\mathbf{g}^T + \left(\mathbf{a}_u^T - \bar{x}_{k^\star}(X)\mathbf{g}^T\right)\mathbf{K}_{w,k^\star,\,\mathsf{low}},$$

  where $\bar{\mathbf{a}}_w \leftarrow\!\$\ \mathcal{R}_q^{m_g}$, $k^\star = \mathsf{LUT}_{\mathcal{L}_w}^{\mathsf{idx}}(z_u(X))$, $z_u(X) \in \mathcal{R}_q$ is a polynomial carried by the input wire u of g_w during the evaluation of C on $\tilde{\mathbf{X}}^\star$, and the definitions of the other variable follow from ones in Algorithm 1.
- ***Hyb 2****: For each query $C \in \mathcal{C}_{\mathcal{R}_q,L,M}$ to KG, corresponding to $C_{\mathcal{R}_q} \in \mathcal{C}_{\mathcal{R}_q,Lw_p,1}$, this hybrid replaces a preimage $\mathbf{K}_{w,k^\star}$ and a vector $\mathbf{a}_w^T$, which is contained by $\mathcal{K}_{C,LUT}$ in dk_C and output by $\mathsf{EvalLut}(\mathbf{a}_u, \mathbf{b}_{\sigma_b}^{-1}, \mathcal{L}_w)$ for every output wire w of the LUT evaluation gate g_w in $C_{\mathcal{R}_q}$, as follows:*

$$\mathbf{K}_{w,k^\star} \leftarrow\!\$\ \mathcal{D}_{\mathcal{R},\sigma_b,\leq\sigma_b\sqrt{\lambda}}^{(m_b+m_g)\times m_g},$$
$$\mathbf{a}_w^T := \mathbf{b}^T\mathbf{K}_{w,k^\star,\,\mathsf{high}} + \bar{y}_{k^\star}(X)\mathbf{g}^T + \left(\mathbf{a}_u^T - \bar{x}_{k^\star}(X)\mathbf{g}^T\right)\mathbf{K}_{w,k^\star,\,\mathsf{low}},$$

  where $\mathbf{K}_{w,k^\star,\,\mathsf{high}} \in \mathcal{R}_q^{m_b\times m_g}$ (resp. $\mathbf{K}_{w,k^\star,\,\mathsf{low}} \in \mathcal{R}_q^{m_g\times m_g}$) is the first m_b rows (resp. the last m_g rows) of $\mathbf{K}_{w,k^\star}$.
- ***Hyb 3****: This hybrid replaces $\mathbf{A}_{\mathsf{inp}} \in \mathcal{R}_q^{m_g\times(1+w_pL)}$ output by $\mathsf{Setup}(1^\lambda)$ as follows:*

$$\mathsf{flatten}(\mathbf{A}_{\mathsf{inp}})^T := \mathbf{b}^T\mathbf{R} + (1, \mathbf{x}_{w_pL}^T) \otimes \mathbf{g}^T,$$

  where $\mathbf{R} \in \mathcal{R}_3^{m_b\times(1+w_pL)m_g}$ and $\mathbf{x}_{w_pL} \in \mathcal{R}_q^{w_pL}$ are derived during the evaluation of Algorithm 6 and $\mathbf{b}$ is output by $\mathsf{Setup}(1^\lambda)$.

– ***Hyb 4****: This hybrid replaces* $(\mathbf{c}_b^T, u(X)) \in \mathcal{R}_q^{1\times(m_b+1)}$ *output by* $\mathsf{Enc}(\mathsf{mpk}, \tilde{\mathbf{X}}^\star, \mu_b)$ *with* $\mathbf{v}^T \leftarrow\!\$\ \mathcal{R}_q^{1\times(m_b+1)}$.

In ***Hyb 4****, the simulated view is completely independent of the challenge bit b, so the adversary's advantage is negligible. Consequently, it remains to prove that each pair of consecutive hybrids is indistinguishable to the adversary except with negligible probability.*

Indistinguishability between Hybs 0 and 1*: For every output wire w of the LUT evaluation gate,* $\mathbf{a}_w$ *remains uniformly distributed across both hybrids. Therefore, the adversary's view is identical between these hybrids.*

Indistinguishability between Hybs 1 and 2*: The only difference between these hybrids is the distributions of* $\mathbf{K}_{w,k^\star,\mathsf{high}}$ *and* $\bar{\mathbf{a}}_w^T$ *for every output wire w of the LUT evaluation gate because the last* m_g *rows of* $\mathbf{K}_{w,k^\star}$ *is already identically distributed in both hybrids. Since the target vector of the preimage* $\mathbf{K}_{w,k^\star}$*, namely* $\mathbf{a}_w^T - \bar{y}_{k^\star}(X)\mathbf{g}^T - \left(\mathbf{a}_u^T - \bar{x}_{k^\star}(X)\mathbf{g}^T\right)\mathbf{K}_{w,k^\star,\mathsf{low}} = \bar{\mathbf{a}}_w^T$ *is uniformly distributed in Hyb 1, we can apply Lemma 2 as below:*

$$\begin{aligned}&\left\{(\mathbf{b},\, \mathbf{K}_{w,k^\star,\mathsf{high}},\, \bar{\mathbf{a}}_w) : \bar{\mathbf{a}}_w \leftarrow\!\$\ \mathcal{R}_q^{m_g}, \mathbf{K}_{w,k^\star,\mathsf{high}} \leftarrow \mathbf{b}_\sigma^{-1}(\bar{\mathbf{a}}_w)\right\}\\ \approx_s &\left\{(\mathbf{b},\, \mathbf{K}_{w,k^\star,\mathsf{high}},\, \bar{\mathbf{a}}_w) : \mathbf{K}_{w,k^\star,\mathsf{high}} \leftarrow\!\$\ \mathcal{D}_{\mathcal{R},\sigma}^{m_b\times m_g}, \bar{\mathbf{a}}_w^T := \mathbf{b}^T\mathbf{K}_{w,k^\star,\mathsf{high}}\right\}.\end{aligned}$$

The LHS and RHS correspond to Hybs 1 and 2, respectively; therefore these hybrids are statistically indistinguishable.

Indistinguishability between Hybs 2 and 3*: The only difference between these hybrids is the distribution of the matrix* $\mathbf{A}_{\mathsf{inp}} \in \mathcal{R}_q^{m_g\times(1+w_pL)}$*. Recall that the vector* $\mathbf{b} \in \mathcal{R}_q^{m_b}$ *output by* $\mathsf{TrapGen}(1^n, 1^{m_b}, 1^q)$ *is* $\mathsf{negl}(n)$*-close to uniform. By applying Lemma 3* $(1 + w_pL)m_g$ *times, we obtain It holds that*

$$\begin{aligned}\left(\mathbf{b}, \mathbf{b}^T\mathbf{R}, \mathbf{e}_b^T\mathbf{R}\right) &\approx_s \left(\mathbf{b}, \mathsf{flatten}(\bar{\mathbf{A}}_{\mathsf{inp}})^T, \mathbf{e}_b^T\mathbf{R}\right)\\ &\approx_s \left(\mathbf{b}, \mathsf{flatten}(\mathbf{A}_{\mathsf{inp}})^T - (1, \mathbf{x}_{w_pL}^T)\otimes\mathbf{g}^T, \mathbf{e}_b^T\mathbf{R}\right),\end{aligned}$$

where $\bar{\mathbf{A}}_{\mathsf{inp}} \leftarrow\!\$\ \mathcal{R}_q^{m_g\times(1+w_pL)}$*. In the above derivation, we obtain the first Indistinguishability by applying Lemma 3* $(1+w_pL)m_g$ *times, and the second one holds because both* $\mathsf{flatten}(\bar{\mathbf{A}}_{\mathsf{inp}})^T$ *and* $\mathsf{flatten}(\mathbf{A}_{\mathsf{inp}})^T - (1, \mathbf{x}_{w_pL}^T)\otimes\mathbf{g}^T$ *are uniformly distributed over* $\mathcal{R}_q^{1\times(1+w_pL)m_g}$*. The first and last distributions correspond to Hybs 3 and 2, respectively, implying that they are statistically indistinguishable.*

Indistinguishability between Hybs 3 and 4*: We show that these hybrids are computationally indistinguishable from each other assuming the ring-LWE. However, to do so we must ensure that the preimages can be sampled without the trapdoor* $\mathbf{b}_{\sigma_b}^{-1}$*.*

For each query $C \in \mathcal{C}_{\mathcal{R}_q,L,M}$ *to* KG*, corresponding to* $C_{\mathcal{R}_q} \in \mathcal{C}_{\mathcal{R}_q,Lw_p,1}$*, we claim that there exists a short matrix* $\mathbf{R}_w \in \mathcal{R}_q^{m_b\times m_g}$ *for every wire w in* $C_{\mathcal{R}_q}$ *such that the following holds:*

$$\mathbf{a}_w^T = \mathbf{b}^T\mathbf{R}_w + z_w(X)\mathbf{g}^T, \tag{11}$$

where $z_w(X) \in \mathcal{R}_q$ is a polynomial carried by the wire w during the evaluation of $C_{\mathcal{R}_q}(\mathbf{x}_{w_pL})$. The replacement performed in Hyb 3 guarantees that Eq. 11 holds for every input wire $u \in [w_pL]$. Specifically, when we parse

$$\begin{aligned}
&\mathsf{flatten}(\mathbf{A}_{\mathsf{inp}})^T = (\mathbf{a}_{\mathsf{inp},\mathsf{one}}^T, \mathbf{a}_{\mathsf{inp},1,1}^T, \ldots, \mathbf{a}_{\mathsf{inp},1,w_p}^T, \ldots, \mathbf{a}_{\mathsf{inp},L,1}^T, \ldots, \mathbf{a}_{\mathsf{inp},L,w_p}^T),\\
&\mathbf{R}_w = (\mathbf{R}_{\mathsf{inp},\mathsf{one}}, \mathbf{R}_{\mathsf{inp},1,1}, \ldots, \mathbf{R}_{\mathsf{inp},1,w_p}, \ldots, \mathbf{R}_{\mathsf{inp},L,1}, \ldots, \mathbf{R}_{\mathsf{inp},L,w_p}),\\
&\mathbf{x}_{w_pL}^T = (z_{\mathsf{inp},1,1}(X), \ldots, z_{\mathsf{inp},1,w_p}(X), \ldots, z_{\mathsf{inp},L,1}(X), \ldots, z_{\mathsf{inp},L,w_p}(X)),
\end{aligned}$$

where $\mathbf{a}_{\mathsf{inp},\mathsf{one}}, \mathbf{a}_{\mathsf{inp},i,j} \in \mathcal{R}_q^{m_g}$ and $\mathbf{R}_{\mathsf{inp},\mathsf{one}}, \mathbf{R}_{\mathsf{inp},i,j} \in \mathcal{R}_q^{m_b \times m_g}$, we have

$$\begin{aligned}
&\mathbf{a}_{\mathsf{inp},\mathsf{one}}^T = \mathbf{b}^T\mathbf{R}_{\mathsf{inp},\mathsf{one}} - \mathbf{g}^T,\\
&\forall i \in [L], j \in [w_p] \quad \mathbf{a}_{\mathsf{inp},i,j}^T = \mathbf{b}^T\mathbf{R}_{\mathsf{inp},i,j} - z_{\mathsf{inp},i,j}(X)\mathbf{g}^T.
\end{aligned}$$

For each gate type defined in Definition 2, we show that Eq. 11 holds for the output w of every gate g_w. Assuming that there are matrices $\mathbf{R}_u, \mathbf{R}_v$ for the inputs u, v to each gate g_w that satisfy Eq. 11, i.e., we define the matrix $\mathbf{R}_w$ in order from the gates in the first layer to those in the last layer as follows.

1. *If g_w is for addition in $\mathcal{R}_q$, let $\mathbf{R}_w := \mathbf{R}_u + \mathbf{R}_v$. It follows that*

$$\begin{aligned}
&\mathbf{b}^T\mathbf{R}_w + z_w(X)\mathbf{g}^T\\
=&\mathbf{b}^T\mathbf{R}_u + z_u(X)\mathbf{g}^T + \mathbf{b}^T\mathbf{R}_v + z_v(X)\mathbf{g}^T\\
=&\mathbf{a}_u^T + \mathbf{a}_v^T\\
=&\mathbf{a}_w^T,
\end{aligned}$$

   where $z_u(X) \in \mathcal{R}_q$ and $z_v(X) \in \mathcal{R}_q$ are polynomials carried by the input wires u and v, respectively.
2. *If g_w is for multiplication in $\mathcal{R}_q$, let $\mathbf{R}_w := \mathbf{R}_u\mathbf{g}^{-1}(\mathbf{a}_v) + z_u(X)\mathbf{R}_v$. It follows that*

$$\begin{aligned}
&\mathbf{b}^T\mathbf{R}_w + z_w(X)\mathbf{g}^T\\
=&\mathbf{b}^T\mathbf{R}_u\mathbf{g}^{-1}(\mathbf{a}_v) + z_u(X)\mathbf{b}^T\mathbf{R}_v + z_w(X)\mathbf{g}^T\\
=&(\mathbf{a}_u^T - z_u(X)\mathbf{g}^T)\mathbf{g}^{-1}(\mathbf{a}_v) + z_u(X)(\mathbf{a}_v^T - z_v(X)\mathbf{g}^T) + z_w(X)\mathbf{g}^T\\
=&\mathbf{a}_u^T\mathbf{g}^{-1}(\mathbf{a}_v) - z_u(X)z_v(X)\mathbf{g}^T + z_w(X)\mathbf{g}^T\\
=&\mathbf{a}_w^T.
\end{aligned}$$

3. *If g_w is for scalar multiplication by $\alpha(X) \in \mathcal{R}_q$, let $\mathbf{R}_w := \mathbf{R}_u\mathbf{g}^{-1}(\alpha(X)\mathbf{g})$. It follows that*

$$\begin{aligned}
&\mathbf{b}^T\mathbf{R}_w + z_w(X)\mathbf{g}^T\\
=&\mathbf{b}^T\mathbf{R}_u\mathbf{g}^{-1}(\alpha(X)\mathbf{g}) + z_w(X)\mathbf{g}^T\\
=&(\mathbf{a}_u^T - z_u(X)\mathbf{g}^T)\mathbf{g}^{-1}(\alpha(X)\mathbf{g}) + z_w(X)\mathbf{g}^T\\
=&\mathbf{a}_u^T\mathbf{g}^{-1}(\alpha(X)\mathbf{g})\\
=&\mathbf{a}_w^T.
\end{aligned}$$

4. *If g_w performs LUT evaluation for a LUT $\mathcal{L}_w \in (\mathcal{R}_q \times \mathcal{R}_q)^{R_w}$, let $\mathbf{S}_w := \mathbf{K}_{w,k^\star,\mathsf{high}} + \mathbf{S}_u \mathbf{K}_{w,k^\star,\mathsf{low}}$, where $k^\star = \mathsf{LUT}^{\mathsf{idx}}_{\mathcal{L}_w}(z_u(X))$ and*

$$\mathbf{K}_{w,k^\star} = \begin{pmatrix} \mathbf{K}_{w,k^\star,\mathsf{high}} \\ \mathbf{K}_{w,k^\star,\mathsf{low}} \end{pmatrix} \leftarrow\!\!\$\ \mathcal{D}^{(m_b+m_g)\times m_g}_{\mathcal{R},\sigma_b,\leq\sigma_b\sqrt{\lambda}}.$$

Recall that $\mathbf{a}_w^T := \mathbf{b}^T\mathbf{K}_{w,k^\star,\mathsf{high}} + \bar{y}_{k^\star}(X)\mathbf{g}^T + \left(\mathbf{a}_u^T - \bar{x}_{k^\star}(X)\mathbf{g}^T\right)\mathbf{K}_{w,k^\star,\mathsf{low}}$ both in Hybs 3 and 4. It follows that

$$\begin{aligned}
&\mathbf{b}^T\mathbf{R}_w + z_w(X)\mathbf{g}^T \\
=&\mathbf{b}^T\mathbf{K}_{w,k^\star,\mathsf{high}} + \mathbf{b}^T\mathbf{S}_u\mathbf{K}_{w,k^\star,\mathsf{low}} + z_w(X)\mathbf{g}^T \\
=&\left(\mathbf{a}_w^T - \bar{y}_{k^\star}(X)\mathbf{g}^T - \left(\mathbf{a}_u^T - z_u(X)\mathbf{g}^T\right)\mathbf{K}_{w,k^\star,\mathsf{low}}\right) + \mathbf{b}^T\mathbf{S}_u\mathbf{K}_{w,k^\star,\mathsf{low}} + z_w(X)\mathbf{g}^T \\
=&\mathbf{a}_w^T - \mathbf{b}^T\mathbf{S}_u\mathbf{K}_{w,k^\star,\mathsf{low}} + \mathbf{b}^T\mathbf{S}_u\mathbf{K}_{w,k^\star,\mathsf{low}} - \bar{y}_{k^\star}(X)\mathbf{g}^T + z_w(X)\mathbf{g}^T \\
=&\mathbf{a}_w^T,
\end{aligned}$$

where $\bar{y}_{k^\star}(X) := \mathsf{LUT}^{\mathsf{out}}_{\mathcal{L}_w}(z_u(X))$, equivalent to $z_w(X)$.

Finally, for each query $C \in \mathcal{C}_{\mathcal{R}_q,L,M}$ to KG, corresponding to $C_{\mathcal{R}_q} \in \mathcal{C}_{\mathcal{R}_q,Lw_p,1}$, we obtain a matrix $\mathbf{R}_{\mathsf{out}} \in \mathcal{R}_q^{m_b\times m_g}$ such that $\mathbf{a}_{\mathsf{out}}^T = \mathbf{b}^T\mathbf{R}_{\mathsf{out}} + y(X)\mathbf{g}^T$ holds, where $y(X) := \mathsf{NTT}^{-1}_{1,\ldots,h}(C(\tilde{\mathbf{X}}^\star))$. Recall that $\mathbf{k}_C \in \mathcal{R}_q^{(m_b+m_g)}$ is a preimage such that $\left(\mathbf{b}^T, \mathbf{a}_{\mathsf{out}}^T\right)\mathbf{k}_C = u(X)$ holds, implying $\left(\mathbf{b}^T, \mathbf{b}^T\mathbf{R}_{\mathsf{out}} + y(X)\mathbf{g}^T\right)\mathbf{k}_C = u(X)$. Since $C(\tilde{\mathbf{X}}^\star) \neq \mathbf{0}_n$ by the definition of KG, $y(X) \neq 0$. Hence, a preimage $\mathbf{k}_C$ output by KG can be simulated without the trapdoor $\mathbf{b}_{\sigma_b}^{-1}$ as follows:

$$\mathbf{k}_C \leftarrow \mathsf{SampLeft}(\mathbf{b}, \mathbf{R}_{\mathsf{out}}, y(X), u(X), \sigma_b),$$

which is $\mathsf{negl}(n)$-close to a coset $\mathcal{D}_{\Lambda^\perp_{u(X)}((\mathbf{b}^T, \mathbf{b}^T\mathbf{R}_{\mathsf{out}} + y(X)\mathbf{g}^T)),\sigma_b}$.

Additionally, for every input wire u of the LUT evaluation gate g_w, there is a matrix $\mathbf{R}_u \in \mathcal{R}_q^{m_b\times m_g}$ such that $\mathbf{a}_u^T = \mathbf{b}^T\mathbf{R}_u + z_u(X)\mathbf{g}^T$ holds. For every $k \neq \mathsf{LUT}^{\mathsf{idx}}_{\mathcal{L}_w}(z_u(X))$, it follows that

$$\begin{aligned}
&\mathbf{a}_u^T - \bar{x}_k(X)\mathbf{g}^T \\
=&\left(\mathbf{b}^T\mathbf{R}_u + z_u(X)\mathbf{g}^T\right) - \bar{x}_k(X)\mathbf{g}^T \\
=&\mathbf{b}^T\mathbf{R}_u + (z_u(X) - \bar{x}_k(X))\mathbf{g}^T.
\end{aligned}$$

By Definition 1, for all $k \neq k' \in [R]$, $\bar{x}_k(X) \neq \bar{x}_{k'}(X)$. Therefore, for any $k \neq \mathsf{LUT}^{\mathsf{idx}}_{\mathcal{L}_w}(z_u(X))$, it holds that $z_u(X) - \bar{x}_k(X) \neq 0$. Consequently, a preimage $\mathbf{K}_{w,k} \in \mathcal{R}_q^{(m_b+m_g)\times m_g}$—satisfying $(\mathbf{b}^T, \mathbf{a}_u^T - \bar{x}_k(X)\mathbf{g}^T)\mathbf{K}_{w,k} = \mathbf{a}_w^T - \bar{y}_k(X)\mathbf{g}^T$—can be simulated without the trapdoor $\mathbf{b}_{\sigma_b}^{-1}$ as follows:

$$\forall k \neq \mathsf{LUT}^{\mathsf{idx}}_{\mathcal{L}_w}(z_u(X)) \quad \mathbf{K}_{w,k} \leftarrow \mathsf{SampLeft}(\mathbf{b}, \mathbf{R}_u, z_u(X) - \bar{x}_k(X), \mathbf{a}_w^T - \bar{y}_k(X)\mathbf{g}^T, \sigma_b),$$

each of which is $\mathsf{negl}(n)$-close to a coset $\mathcal{D}_{\Lambda^\perp_{\mathbf{a}_w^T - \bar{y}_k(X)\mathbf{g}^T}((\mathbf{b}^T, \mathbf{b}^T\mathbf{R}_u + (z_u(X) - \bar{x}_k(X))\mathbf{g}^T)),\sigma_b}$. Hence, all preimages can be simulated without the trapdoor.

We finally prove the computational indistinguishability between Hybs 3 and 4 assuming the ring-LWE. Suppose an adversary $\mathcal{A}$ that can distinguish between them with a non-negligible advantage, an adversary $\mathcal{B}$ can be constructed that can break the hardness of the ring-LWE in Assumption 1 by internally invoking $\mathcal{A}$.

$\mathcal{B}$ receives a challenge $(\bar{\mathbf{b}}, \bar{\mathbf{c}}) \in \mathcal{R}_q^{m_b+1} \times \mathcal{R}_q^{m_b+1}$ from a challenger of the RLWE security game, where $\bar{\mathbf{b}} \leftarrow\!\!\$\ \mathcal{R}_q^{m+1}$, and $\bar{\mathbf{c}}$ is either $\bar{\mathbf{c}}^T = s(X)\bar{\mathbf{b}}^T + \bar{\mathbf{e}}^T$ for $s(X) \leftarrow\!\!\$\ \mathcal{R}_3$ and $\bar{\mathbf{e}} \leftarrow\!\!\$\ \mathcal{D}^{m_b+1}_{\mathcal{R},\sigma_e,\leq\sigma_e\sqrt{\lambda}}$ in the real case or $\bar{\mathbf{c}} \leftarrow\!\!\$\ \mathcal{R}_q^{m+1}$ in the ideal case. $\mathcal{B}$ parses $\bar{\mathbf{b}}$ and $\bar{\mathbf{c}}$ as follows:

$$\bar{\mathbf{b}}^T = (\mathbf{b}^T, u(X)),$$
$$\bar{\mathbf{c}}^T = (\mathbf{c}_b^T, c_u(X)),$$

where $\mathbf{c}_b^T = s(X)\mathbf{b}^T + \mathbf{e}_b^T$ and $c_u(X) = s(X)u(X) + e_u(X)$ in the real case.

$\mathcal{A}$ first provides a challenge attribute $\tilde{\mathbf{X}}^\star$. $\mathcal{B}$ provides $\mathcal{A}$ a master public key mpk that contains the given $\mathbf{b}^T$ and $u(X)$ along with $\mathbf{A}_{\mathsf{inp}} \in \mathcal{R}_q^{m_g \times (1+w_pL)}$ such that $\mathsf{flatten}(\mathbf{A}_{\mathsf{inp}})^T := \mathbf{b}^T\mathbf{R} + (1, \mathbf{x}_{w_pL}^T) \otimes \mathbf{g}^T$. For each $\mathcal{A}$'s query $C \in \mathcal{C}_{\mathcal{R}_q,L,M}$ to KG, $\mathcal{B}$ can simulate the output decryption key dk_C without the trapdoor $\mathbf{b}_{\sigma_b}^{-1}$ as shown above.

$\mathcal{A}$ then returns challenge messages $\mu_0, \mu_1 \in \{0,1\}$ to $\mathcal{B}$. $\mathcal{B}$ constructs a ciphertext ct by employing the given $(\mathbf{c}_b^T, c_u(X))$ and $\mathbf{c}_a \in \mathcal{R}_q^{(1+w_pL)m_g}$ defined by $\mathbf{c}_a^T := \mathbf{c}_b^T\mathbf{R}$. In the real case of the RLWE security game, ct corresponds to one in Hyb 3 because it holds that

$$\begin{aligned}
\mathbf{c}_a =& \mathbf{c}_b^T\mathbf{R} \\
=& \left(s(X)\mathbf{b}^T + \mathbf{e}_b^T\right)\mathbf{R} \\
=& s(X)\mathbf{b}^T\mathbf{R} + \mathbf{e}_b^T\mathbf{R} \\
=& s(X)\left(\mathsf{flatten}(\mathbf{A}_{\mathsf{inp}})^T - (1, \mathbf{x}_{w_pL}^T) \otimes \mathbf{g}^T\right) + \mathbf{e}_b^T\mathbf{R}.
\end{aligned}$$

In the ideal case of the RLWE security game, ct corresponds to one in Hyb 4 because $\mathbf{c}_b^T\mathbf{R}$ is statistically indistinguishable from a uniformly random vector by Lemma 3. Hence, by providing ct, $\mathcal{B}$ can simulate the $\mathcal{A}$'s view in both hybrids.

$\mathcal{B}$ finally forwards b' output by $\mathcal{A}$ to the RLWE challenger. Since $b = b'$ with a non-trivial probability, $\mathcal{B}$ also distinguishes between the real and ideal cases of the RLWE challenge. By contradiction, we can conclude that assuming the hardness of RLWE, Hybs 3 and 4 are computationally indistinguishable from each other. □

7 Implementation and Evaluation

8 Conclusion

References

1. Agrawal, S., Boneh, D., Boyen, X.: Efficient lattice (h) ibe in the standard model. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 553–572. Springer (2010)
2. Ajtai, M.: Generating hard instances of the short basis problem. In: International Colloquium on Automata, Languages and Programming (1999), `https://api.semanticscholar.org/CorpusID:29824140`
3. Alamati, N., Peikert, C.: Three's compromised too: Circular insecurity for any cycle length from (ring-) lwe. In: Annual International Cryptology Conference. pp. 659–680. Springer (2016)
4. Albrecht, M., Chase, M., Chen, H., Ding, J., Goldwasser, S., Gorbunov, S., Halevi, S., Hoffstein, J., Laine, K., Lauter, K., et al.: Homomorphic encryption standard. In: Protecting privacy through homomorphic encryption, pp. 31–62. Springer (2022)
5. Alwen, J., Peikert, C.: Generating shorter bases for hard random lattices. In: 26th International Symposium on Theoretical Aspects of Computer Science (2009). pp. 75–86. Schloss Dagstuhl–Leibniz-Zentrum für Informatik (2009)
6. Bolboceanu, M., Costache, A., Hales, E., Player, R., Rosca, M., Titiu, R.: Designs for practical she schemes based on ring-lwr. Cryptology ePrint Archive (2024)
7. Boneh, D., Gentry, C., Gorbunov, S., Halevi, S., Nikolaenko, V., Segev, G., Vaikuntanathan, V., Vinayagamurthy, D.: Fully key-homomorphic encryption, arithmetic circuit abe and compact garbled circuits. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 533–556. Springer (2014)
8. Dai, W., Doröz, Y., Polyakov, Y., Rohloff, K., Sajjadpour, H., Savaş, E., Sunar, B.: Implementation and evaluation of a lattice-based key-policy abe scheme. IEEE Transactions on Information Forensics and Security **13**(5), 1169–1184 (2017)
9. Dodis, Y., Ostrovsky, R., Reyzin, L., Smith, A.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. SIAM journal on computing **38**(1), 97–139 (2008)
10. Gentry, C., Halevi, S., Smart, N.P.: Homomorphic evaluation of the aes circuit. In: Annual Cryptology Conference. pp. 850–867. Springer (2012)
11. Gentry, C., Peikert, C., Vaikuntanathan, V.: How to use a short basis: Trapdoors for hard lattices and new cryptographic constructions. In: Proceedings of the 40th Annual ACM Symposium on Theory of Computing. pp. 197–206 (2009)
12. Halevi, S., Polyakov, Y., Shoup, V.: An improved rns variant of the bfv homomorphic encryption scheme. In: Cryptographers' Track at the RSA Conference. pp. 83–105. Springer (2019)
13. Hsieh, Y.C., Lin, H., Luo, J.: Attribute-based encryption for circuits of unbounded depth from lattices: garbled circuits of optimal size, laconic functional evaluation, and more. SIAM Journal on Computing pp. FOCS23–90 (2025)
14. Kogge, P.M., Stone, H.S.: A parallel algorithm for the efficient solution of a general class of recurrence equations. IEEE transactions on computers **100**(8), 786–793 (2009)

15. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. Journal of the ACM (JACM) **60**(6), 1–35 (2013)
16. Menezes, A.J., Van Oorschot, P.C., Vanstone, S.A.: Handbook of applied cryptography. CRC press (2018)
17. Micciancio, D., Peikert, C.: Trapdoors for lattices: Simpler, tighter, faster, smaller. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 700–718. Springer (2012)
18. Wallace, C.S.: A suggestion for a fast multiplier. IEEE Transactions on electronic Computers (1), 14–17 (2006)

A Arithmetic

This section illustrates the algorithms to perform modular arithmetic operations. We then analyse the depth of the corresponding arithmetic circuits.

The inputs to the following algorithms are vectors of polynomials in $\mathcal{R}_q^{w_p}$, where each vector represents the base-B_p limbs of the input integer vector in $\mathcal{Z}_{q_i}^n$ placed in the evaluation slots of the polynomials. By the ring-homomorphism property of the evaluation slots, all ring operations over $\mathcal{R}_q$ correspond to component-wise arithmetics over $\mathbb{Z}_q^n$. Consequently, for any $a(X), b(X) \in \mathcal{R}_q$, it holds that

$$\begin{aligned} a(X) + b(X) &= \mathsf{NTT}_{1,\ldots,h}^{-1}(\tilde{\mathbf{a}} \boxplus \tilde{\mathbf{b}}), \\ a(X) - b(X) &= \mathsf{NTT}_{1,\ldots,h}^{-1}(\tilde{\mathbf{a}} \boxminus \tilde{\mathbf{b}}), \\ a(X)b(X) &= \mathsf{NTT}_{1,\ldots,h}^{-1}(\tilde{\mathbf{a}} \boxtimes \tilde{\mathbf{b}}), \end{aligned}$$

where $\tilde{\mathbf{a}} := \mathsf{NTT}_{1,\ldots,h}(a(X))$ and $\tilde{\mathbf{b}} := \mathsf{NTT}_{1,\ldots,h}(b(X))$.

A.1 Multi-Precision Addition and Subtraction

We first instantiate multi-precision addition and subtraction circuits $C_{+,i}, C_{-,i} \in \mathcal{C}_{\mathcal{R}_q, 2w_p, w_p+1}$ described by Algorithms 9 and 10, respectively. The inputs to these circuits are two vectors $\mathbf{a}, \mathbf{b} \in \mathcal{R}_q^{w_p}$. The former circuit outputs a vector in $\mathcal{R}_q^{w_p+1}$ that corresponds to the $w_p + 1$ limbs of the sum obtained when treating $\tilde{\mathbf{a}}$ and $\tilde{\mathbf{b}}$ respectively as big integers. The latter circuit outputs a vector in $\mathcal{R}_q^{w_p}$ along with one more polynomial in $\mathcal{R}_q$ such that the vector corresponds to limbs of the difference of these big integers, and the last polynomial represents borrow-out bits in the evaluation slots. In both circuits, polynomials representing per-limb generate/propagate bits, denoted by $g_l(X), p_l(X)$ for $l \in [w_p]$, are obtained by Algorithms 3 and 11,

Lemma 9. *We keep the same parameter settings in Definition 4. For every $i \in [h]$, two circuits $C_{+,i}, C_{-,i}$, implementing, respectively Algorithm 9 and Algorithm 10, belong to the class $\mathcal{C}_{\mathcal{R}_q, 2w_p, w_p+1}$ and satisfy the following properties:*

Algorithm 9 $\mathsf{SIMDAdd}_{q,n,q_i,B_p}$

Parameter Settings: $q_i \mid q$, $(B_p-1)^2 < q_i$ and $w_p = \lceil \log_{B_p} q_i \rceil$.
Input: $\mathbf{a} := (a_1(X), \ldots, a_{w_p}(X))^T \in \mathcal{R}_q^{w_p}$, $\mathbf{b} := (b_1(X), \ldots, b_{w_p}(X))^T \in \mathcal{R}_q^{w_p}$.
Output: $\mathbf{c} \in \mathcal{R}_q^{w_p+1}$

for $l \in [w_p]$ **do**
  $t_l(X) := a_l(X) + b_l(X)$.
  $s_l(X), g_l(X) \leftarrow \mathsf{Mod\&Floor}_{q,n,q_i,B_p}(t_l(X))$.
  $_, p_l(X) \leftarrow \mathsf{Mod\&Floor}_{q,n,q_i,B_p}(s_l(X)+1)$.
end for
$\mathbf{g} := (g_l(X))_{l\in[w_p]}^T$, $\mathbf{p} := (p_l(X))_{l\in[w_p]}^T$.
$\mathbf{g}' \leftarrow \mathsf{KoggeStonePrefixSum}_q(\mathbf{g}, \mathbf{p})$.
$\mathbf{s}' := (0, g'_1(X), \ldots, g'_{w_p-1}(X))^T \in \mathcal{R}_q^{w_p}$.
for $l \in [w_p]$ **do**
  $c_l(X), _ \leftarrow \mathsf{Mod\&Floor}_{q,n,q_i,B_p}(s_l(X) + s'_l(X))$.
end for
$c_{w_p+1}(X), _ \leftarrow \mathsf{Mod\&Floor}_{q,n,q_i,B_p}(g'_{w_p}(X))$.
$\mathbf{c} := (c_1(X), \ldots, c_{w_p}(X), c_{w_p+1}(X))^T$.
return $\mathbf{c}$.

1. *The below set is (a subset of) the admissible input sets of* $C_{\pm,i}$*:*

$$\mathcal{X}_{C_{\pm,i},B_\times,C_{\pm,i}} := \left\{ \begin{pmatrix} \left(\mathsf{NTT}_i^{-1}(\tilde{\mathbf{a}}_l) \mod q\mathcal{R} + q_i e_{a_l}(X)\right)_{l\in[w_p]}, \\ \left(\mathsf{NTT}_i^{-1}\left(\tilde{\mathbf{b}}_l\right) \mod q\mathcal{R} + q_i e_{b_l}(X)\right)_{l\in[w_p]} \end{pmatrix} \right\},$$

   where $\tilde{\mathbf{a}}_l, \tilde{\mathbf{b}}_l \in [0, B_p-1]^n$*,* $e_{a_l}(X)$ *and* $e_{b_l}(X)$ *are polynomials in* $\mathcal{R}_q$ *such that* $|e_{a_l}(X)|, |e_{b_l}(X)| \le n-1$*, and* $B_{\times,C_{\pm,i}} = (n(q_i-1))^{3^{\mathcal{O}(\lceil \log_2 w_p \rceil)}}$.
2. *The non-free depth of* $C_{+,i}$ *is bounded by* $2\lceil \log_2 w_p \rceil + 8$*. The non-free depth of* $C_{-,i}$ *is bounded by* $2\lceil \log_2 w_p \rceil + 9$*. The additive width of both circuits is* n*.*
3. *For any vector* $\begin{pmatrix} (a_l(X))_{l\in[w_p]} \\ (b_l(X))_{l\in[w_p]} \end{pmatrix} \in \mathcal{X}_{C_{\pm,i},B_{\times,q_i}}$*, there exist* $\mathbf{e}_c \in \mathcal{R}_q^{w_p+1}$ *such that*

$$\mathbf{c} = \left(\mathsf{NTT}_i^{-1}\left(\left[\tilde{\mathbf{c}}_l\right]_{q_i}\right) \mod q\mathcal{R}\right)_{l\in[w_p+1]}^T + q_i\mathbf{e}_c,$$
$$\|\mathbf{e}_c\|_\infty \le n-1$$

Algorithm 10 $\mathsf{SIMDSub}_{q,n,q_i,B_p}$

Parameter Settings: $q_i \mid q$, $(B_p-1)^2 < q_i$ and $w_p = \lceil \log_{B_p} q_i \rceil$.
Input: $\mathbf{a} := (a_1(X), \ldots, a_{w_p}(X))^T \in \mathcal{R}_q^{w_p}$, $\mathbf{b} := (b_1(X), \ldots, b_{w_p}(X))^T \in \mathcal{R}_q^{w_p}$.
Output: $\mathbf{d} \in \mathcal{R}_q^{w_p}$, $c(X) \in \mathcal{R}_q$

for $l \in [w_p]$ **do**
  $t_l(X) := a_l(X) + (B_p - 1) - b_l(X)$.
  $s_l(X), h_l(X) \leftarrow \mathsf{Mod\&Floor}_{q,n,q_i,B_p}(t_l(X))$.
  $_, p_l(X) \leftarrow \mathsf{Mod\&Floor}_{q,n,q_i,B_p}(s_l(X) + 1)$.
  $g_l(X) := (1 - h_l(X))(1 - p_l(X))$.
end for
$\mathbf{g} := (g_l(X))_{l\in[w_p]}^T$, $\mathbf{p} := (p_l(X))_{l\in[w_p]}^T$.
$\mathbf{g}' \leftarrow \mathsf{KoggeStonePrefixSum}_q(\mathbf{g}, \mathbf{p})$.
$\mathbf{s}' := (0, g'_1(X), \ldots, g'_{w_p-1}(X))^T \in \mathcal{R}_q^{w_p}$.
for $l \in [w_p]$ **do**
  $d_l(X), _ \leftarrow \mathsf{Mod\&Floor}_{q,n,q_i,B_p}(a_l(X) + B_p - b_l(X) - s'_l(X))$.
end for
$\mathbf{d} := (d_1(X), \ldots, d_{w_p}(X))^T$.
$c(X), _ \leftarrow \mathsf{Mod\&Floor}_{q,n,q_i,B_p}(g'_{w_p}(X))$. ▷ borrow-out polynomial
return $\mathbf{d}, c(X)$.

hold, where

$$\forall l \in [w_p] \quad \tilde{\mathbf{a}}_l := \mathsf{NTT}_i(a_l(X) \mod q_i\mathcal{R}) \in \mathbb{Z}_{q_i}^n,$$
$$\forall l \in [w_p] \quad \tilde{\mathbf{b}}_l := \mathsf{NTT}_i(b_l(X) \mod q_i\mathcal{R}) \in \mathbb{Z}_{q_i}^n,$$

$$\mathbf{c} := C_{+,i}\begin{pmatrix}(a_l(X))_{l\in[w_p]}\\(b_l(X))_{l\in[w_p]}\end{pmatrix},$$
$$\left([\tilde{\mathbf{c}}_l]_{q_i}\right)_{l\in[w_p+1]}^T \in \mathbb{Z}_q^{n\times(w_p+1)}$$
s.t. $\boxplus_{l\in[w_p+1]} B_p^{l-1}\tilde{\mathbf{c}}_l = \left(\boxplus_{l\in[w_p]} B_p^{l-1}\tilde{\mathbf{a}}_l\right) \boxplus \left(\boxplus_{l\in[w_p]} B_p^{l-1}\tilde{\mathbf{b}}_l\right)$ *over integers.*

4. *For any vector* $\begin{pmatrix}(a_l(X))_{l\in[w_p]}\\(b_l(X))_{l\in[w_p]}\end{pmatrix} \in \mathcal{X}_{C_{\pm,i},B_{\times},q_i}$, *there exist* $\mathbf{e}_d \in \mathcal{R}_q^{w_p}$ *and* $e_c(X) \in \mathcal{R}_q$ *such that*

$$\mathbf{d} = \left(\mathsf{NTT}_i^{-1}\left(\left[\tilde{\mathbf{d}}_l\right]_{q_i}\right) \mod q\mathcal{R}\right)_{l\in[w_p]}^T + q_i\mathbf{e}_d,$$
$$\|\mathbf{e}_d\|_\infty \le n-1,$$
$$c(X) = \mathsf{NTT}_i^{-1}\left([\tilde{\mathbf{c}}]_{q_i}\right) \mod q\mathcal{R} + q_i e_c(X),$$
$$|e_c(X)| \le n-1$$

Algorithm 11 $\mathsf{SIMDKoggeStonePrefixSum}_q$

Input: $\mathbf{g} = (g_1(X), \ldots, g_{w_p}(X))^T \in \mathcal{R}_q^{w_p}$, $\mathbf{p} = (p_1(X), \ldots, p_{w_p}(X))^T \in \mathcal{R}_q^{w_p}$.
Output: $\mathbf{g} \in \mathcal{R}_q^{w_p}$.

$d \leftarrow 1$.
while $d < w_p$ **do**
  for $l \in [d+1, w_p]$ **do**
    $r_l(X) := p_l(X) g_{l-d}(X)$.
    $g'_l(X) := g_l(X) + r_l(X) - g_l(X) r_l(X)$.
    $p'_l(X) := p_l(X) p_{l-d}(X)$.
  end for
  $d \leftarrow 2d$.
  $\mathbf{g} \leftarrow (g_1(X), \ldots, g_d(X), g'_{d+1}(X), \ldots, g'_{w_p}(X))$.
  $\mathbf{p} \leftarrow (p_1(X), \ldots, p_d(X), p'_{d+1}(X), \ldots, p'_{w_p}(X))$.
end while
return g

hold, where

$$\mathbf{d}, c(X) := C_{-,i}\begin{pmatrix}(a_l(X))_{l\in[w_p]}\\(b_l(X))_{l\in[w_p]}\end{pmatrix},$$

$$\left(\left[\tilde{\mathbf{d}}_l\right]_{q_i}\right)^T_{l\in[w_p]} \in \mathbb{Z}_q^{n\times w_p}$$

s.t. $\boxplus_{l\in[w_p]} B_p^{l-1}\tilde{\mathbf{d}}_l = \left(\boxplus_{l\in[w_p]} B_p^{l-1}\tilde{\mathbf{a}}_l\right) \boxminus \left(\boxplus_{l\in[w_p]} B_p^{l-1}\tilde{\mathbf{b}}_l\right)$ *over integers,*

and $\tilde{\mathbf{c}} \in \{0,1\}^n$ *such that, for every* $j \in [n]$*, the* j*-th element* c_j *is 1 if the* j*-th element of* $\left(\boxplus_{l\in[w_p]} B_p^{l-1}\tilde{\mathbf{a}}_l\right)$ *is less than that of* $\left(\boxplus_{l\in[w_p]} B_p^{l-1}\tilde{\mathbf{b}}_l\right)$ *and 0 otherwise.*

Proof. We prove the claim for the addition circuit $C_{+,i}$*; the proof for* $C_{-,i}$ *is analogous with the same depth and admissibility analysis. We decompose the process in Algorithm 9 into two stages.*

Stage 1: Per-limb addition.

Functionality: *For every* $l \in [w_p]$*, let* $t_l(X) = a_l(X) + b_l(X)$ *and executes* $\mathsf{Mod\&Floor}_{q,n,q_i,B_p}$ *(Algorithm 3). By Lemma 6, it follows that the outputs* $s_l(X)$ *and* $g_l(X)$ *correspond to the digit and carry of the integer in the* l*-th evaluation slot of* $t_l(X)$*, respectively, as follows:*

$$s_l(X) = \mathsf{NTT}_i^{-1}\left((\tilde{t}_{l,j} \mod B_p)_{j\in[n]}\right) \mod q\mathcal{R} + q_i e_{s_l}(X),$$
$$g_l(X) = \mathsf{NTT}_i^{-1}\left(\left(\left\lfloor\frac{\tilde{t}_{l,j}}{B_p}\right\rfloor\right)_{j\in[n]}\right) \mod q\mathcal{R} + q_i e_{g_l}(X),$$

where $(\tilde{t}_{l,j})_{j\in[n]} := \mathsf{NTT}_i([t_l(X)]_{q_i})$ *and* $|e_{s_l}(X)|, |e_{g_l}(X)| \le n-1$*. The* $\mathsf{Mod\&Floor}_{q,n,q_i,B_p}$ *algorithm is called again to obtain* $p_l(X)$ *that represents bits* $(\tilde{p}_{l,j})_{j\in[n]} \in \{0,1\}^n$

in its evaluation slots such that $\tilde{p}_{l,j}$ *is 1 if* $\tilde{t}_{l,j} \mod B_p$ *is equal to* $B_p - 1$ *and 0 otherwise. Since* $\left\lfloor \frac{\tilde{t}_{l,j} \mod B_p + 1}{B_p} \right\rfloor = 1$ *if and only if* $\tilde{p}_{l,j} = 1$*, we can compute*

$$p_l(X) = \mathsf{NTT}_i^{-1}\left((\left\lfloor \frac{\tilde{t}_{l,j} \mod B_p + 1}{B_p} \right\rfloor)_{j\in[n]} \right) + q_i e_{p_l}(X),$$

where $|e_{p_l}(X)| \le n-1$.

Input admissibility: *The definition of* $\mathcal{X}_{C_{\pm,i},B_{\times,q_i}}$ *ensures that* $[\tilde{\mathbf{a}}_l]_{q_i}, \left[\tilde{\mathbf{b}}_l\right]_{q_i} \in [0, B_p-1]^n$ *for every* $l \in [w_p]$*. By the ring-homomorphic property, it follows that*

$$t_l(X) = \mathsf{NTT}_i^{-1}([\tilde{\mathbf{a}}_l]_{q_i} \boxplus \left[\tilde{\mathbf{b}}_l\right]_{q_i}) \mod q\mathcal{R} + q_i e_{t_l}(X),$$

where $|e_{t_l}(X)| \le |e_{a_l}(X)| + |e_{b_l}(X)| + 1$*. We can confirm that* $t_l(X)$ *belongs to the admissible input set of* $C_{\mathsf{Mod\&Floor},i}$ *defined in Lemma 6 because* $\left\| [\tilde{\mathbf{a}}_l]_{q_i} \boxplus \left[\tilde{\mathbf{b}}_l\right]_{q_i} \right\|_\infty \le 2(B_p-1) \le (B_p-1)^2$*; hence the first call to* Mod&Floor *in Algorithm 9 never returns* $\perp$*. Similarly, since* $(B_p-1)+1 \le (B_p-1)^2$*,* $s_l(X)+1$ *also belongs to that admissible input set so that the second call does not return* $\perp$*.*

Bound of the left-hand inputs to multiplication gates: *This stage involves two additions and two calls to* Mod&Floor *for every* $l \in [w_p]$*. Since the bound of the left-hand inputs to multiplication gates in* $C_{\mathsf{Mod\&Floor},i}$ *is zero as shown in Lemma 6, this bound in this stage is* $B_{\times, C_{\pm,i},1} = 0$*.*

Non-free depth and additive width: *We can easily confirm that the additive width in this stage is same as that of* $C_{\mathsf{Mod\&Floor},i}$*, i.e.,* n*. The second call to* Mod&Floor *depends on* $s_l(X)$ *output by the first call to* Mod&Floor*. Therefore, the non-free depth in this stage is twice the non-free depth of* Mod&Floor*, namely* 4*.*

Stage 2: Kogge–Stone prefix sum.

Functionality: *The inputs* $\mathbf{g}, \mathbf{p}$ *to Algorithm 11 are assumed to encode bits with some errors multiplied by* q_i *in their evaluation slots, i.e.,*

$$\forall l \in [w_p] \quad g_l(X) = \mathsf{NTT}_i^{-1}(\tilde{\mathbf{g}}_l) \mod q\mathcal{R} + q_i e_{g_l}(X),$$
$$\forall l \in [w_p] \quad p_l(X) = \mathsf{NTT}_i^{-1}(\tilde{\mathbf{p}}_l) \mod q\mathcal{R} + q_i e_{p_l}(X)$$

for some $\tilde{\mathbf{g}}_l, \tilde{\mathbf{p}}_l \in \{0,1\}^n$ *and* $e_{g_l}(X), e_{p_l}(X) \in \mathcal{R}_q$ *such that* $|e_{g_l}(X)|, |e_{p_l}(X)| \le n-1$*. The call to this algorithm in Algorithm 9 satisfies this condition because* $\left\lfloor \frac{x}{B_p} \right\rfloor \le 1$ *if* $[x]_{q_i} \le 2B_p - 1$*, which is true for* $g_l(X)$ *and* $p_l(X)$ *computed in Algorithm 9.*

Lines 4–6 in Algorithm 11 perform binary operations in the evaluation slots as follows:

- *Line 4 computes a binary AND operation for each evaluation slot, i.e.,*

$$r_l(X) = \mathsf{NTT}_i^{-1}\left((\tilde{p}_{l,j} \wedge \tilde{g}_{l-d,j})_{j\in[n]} \right) \mod q\mathcal{R} + q_i e_{r_l}(X),$$

where $\tilde{\mathbf{g}}_l = (\tilde{g}_{l,j})_{j\in[n]}$, $\tilde{\mathbf{p}}_l = (\tilde{p}_{l,j})_{j\in[n]}$, *and* $e_{r_l}(X) \in \mathcal{R}_q$. *It holds because* $\tilde{p}_{l,j} \wedge \tilde{g}_{l-d,j} = \tilde{p}_{l,j}\tilde{g}_{l-d,j} \mod q_i$ *as* $\tilde{p}_{l,j}, \tilde{g}_{l-d,j} \in \{0,1\}$.

– *Line 5 computes a binary OR operation for each evaluation slot, i.e.,*

$$g'_l(X) = \mathsf{NTT}_i^{-1}\left((\tilde{g}_{l,j} \vee \tilde{r}_{l,j})_{j\in[n]}\right) \mod q\mathcal{R} + q_i e_{g'_l}(X),$$

where $e_{g'_l}(X) \in \mathcal{R}_q$. *It holds because* $\tilde{g}_{l,j} \vee \tilde{r}_{l,j} = \tilde{g}_{l,j} + \tilde{r}_{l,j} - \tilde{g}_{l,j}\tilde{r}_{l,j} \mod q_i$ *as* $\tilde{g}_{l,j}, \tilde{r}_{l,j} \in \{0,1\}$.

– *Line 6 computes a binary AND operation in the evaluation slots again as follows:*

$$p'_l(X) = \mathsf{NTT}_i^{-1}\left((\tilde{p}_{l,j} \wedge \tilde{p}_{l-d,j})_{j\in[n]}\right) \mod q\mathcal{R} + q_i e_{p'_l}(X),$$

where $e_{p'_l}(X) \in \mathcal{R}_q$.

The correspondence established above between the arithmetic over $\mathcal{R}_q$ *and binary operations shows that, in the evaluation slots of polynomial, Algorithm 11 carries out essentially the same computation as the textbook Kogge–Stone prefix sum algorithm [14]. Therefore, in Algorithm 9, for every* $l \in [2, w_p]$ *and* $j \in [n]$, *the* j*-th evaluation slot of* $g'_{l-1}(X)$ *represents the carry that should be added to the* l*-th limb of the sum* $t_l(X)$, *i.e.,* $s_l(X)$. *Additionally,* $g'_{w_p}(X)$ *represents the carry of the addition in the evaluation slots. A polynomial* $c_{w_p+1}(X)$ *output by* $\mathsf{Mod\&Floor}_{q,n,q_i,B_p}(g'_{w_p}(X))$ *encode the same bits as ones of* $g'_{w_p}(X)$ *in the evaluation slots; however, this reduces the norm of the error polynomial multipled by* q_i. *Hence, the output* $\mathbf{c}$ *satisfies*

$$\mathbf{c} = \left(\mathsf{NTT}_i^{-1}\left([\tilde{\mathbf{c}}_l]_{q_i}\right) \mod q\mathcal{R}\right)_{l\in[w_p+1]}^{T} + q_i\mathbf{e}_c$$

with some $\mathbf{e}_c \in \mathcal{R}_q^{w_p}$. *Since all polynomials in* $\mathbf{c}$ *are the outputs of Algorithm 3,* $\|\mathbf{e}_c\|_\infty \le n-1$ *holds from Lemma 6.*

Input admissibility: *Recall that* $s_l(X)$ *and* $s'_l(X)$ *encodes integers less than* B_p *and bits in the evaluation slots, respectively. For every* $l \in [w_p]$, *it follows that*

$$\left\|\mathsf{NTT}_i([s_l(X) + s'_l(X)]_{q_i})\right\|_\infty \le (B_p - 1) + 1 \le (B_p - 1)^2.$$

Therefore, the input to $\mathsf{Mod\&Floor}_{q,n,q_i,B_p}(s_l(X) + s'_l(X))$ *at Line 10 in Algorithm 9 belongs to the admissible input set of* $C_{\mathsf{Mod\&Floor},i}$ *defined in Lemma 6. The same holds for the input to* $\mathsf{Mod\&Floor}_{q,n,q_i,B_p}(g'_{w_p}(X))$. *Hence, no LUT evaluation gate returns* $\perp$ *in the second stage.*

Bound of the left-hand inputs to multiplication gates: *Within the second stage, only Lines 4–6 in Algorithm 11 use polynomial-to-polynomial multiplication. Let* $k_{\max}$ *be the maximum integer such that* $2^k < w_p$ *holds and* $d_{\max} := 2^{k_{\max}}$. *The left-hand input to the multiplication gate is either* $g_l(X)$ *or* $g_l(X)$ *for each* $d \in \{1, 2, \ldots, d_{\max}\}$ *and* $l \in [d+1, w_p]$. *When* $d = 1$, *by the definitions of* $\mathbf{g}$ *and* $\mathbf{p}$ *output by the first stage, for all* $l \in [2, w_p]$, *it follows that*

$$|g_l(X)|, |p_l(X)| \le n(q_i - 1).$$

Suppose that for each $k \in [k_{max}-1]$, there exists an integer B_k such that $|g_l(X)|, |p_l(X)| \leq B_k$ for $d = 2^k$ and all $l \in [w_p]$, e.g., $B_0 = n(q_i - 1)$ as shown above. When $d = 2^{k+1}$, $|r_l(X)| \leq nB_k^2$ holds. We can define $B_{k+1} = 3n^2B_k^3$ as follows:

$$\begin{aligned}
&\max\{|g'_l(X)|, |p'_l(X)|\} \\
=&|g'_l(X)| \\
\leq&|g_l(X)| + |r_l(X)| + n|g_l(X)|\,|r_l(X)| \\
\leq&B_k + nB_k^2 + n^2B_k^3 \\
\leq&(1+n)B_k^3 + n^2B_k^3 \\
\leq&2nB_k^3 + n^2B_k^3 \\
\leq&3n^2B_k^3 \\
=&B_{k+1}.
\end{aligned}$$

The general solution of B_k is $B_k := (3n^2)^{\frac{3^k-1}{2}}(n(q_i-1))^{3^k}$. As $k_{\max} < \lceil \log_2 w_p \rceil$, the bound of the left-hand inputs to all multiplication gates in the second stage is $B_{\times,C_{\pm,i},2} = (n(q_i-1))^{3^{\mathcal{O}(\lceil \log_2 w_p \rceil)}}$.

Non-free depth and additive width: *Recall that the non-free depth of $C_{\mathsf{Mod\&Floor},i}$ is 2 (Lemma 6). That of the circuit corresponding to Algorithm 11 is bounded by $2\lceil \log_2 w_p \rceil$. Since there is no other non-free gate in Algorithm 9, the non-free depth in the second stage is bounded by $2(\lceil \log_2 w_p \rceil)$. Two additional calls to* Mod&Floor*, where the former is run in parallel for every $l \in [w_p]$, increase the non-free depth by 4 within Algorithm 9. Therefore, the non-free depth in the second stage is $2\lceil \log_2 w_p \rceil + 4$. The additive width in the second stage is n, which comes from that of $C_{\mathsf{Mod\&Floor},i}$.*

Properties of $C_{+,i}$.
We prove that the three properties of $C_{+,i}$ hold. Recall that the first property claims the admissible input set of $C_{+,i}$. By the input admissibilities of the first and second stages, we can immediatly show that no LUT evaluation gate returns $\perp$. It also follows that $B_{\times,C_{\pm,i}} = \max\{B_{\times,C_{\pm,i},1}, B_{\times,C_{\pm,i},2}\} = (n(q_i - 1))^{3^{\mathcal{O}(\lceil \log_2 w_p \rceil)}}$. The non-free depth in the second property is obtained by adding non-free depths in two stages, resulting in $2\lceil \log_2 w_p \rceil + 8$, and the additive width is n. The third property—the expected functionality of $C_{+,i}$—is also derived by combining the functionalities in the first and second stages.

Properties of $C_{-,i}$.
The non-free depth of the subtraction circuit $C_{-,i}$ accounts for an additional multiplication gate in the first step of Algorithm 10, therefore resulting in $2\lceil \log_2 w_p \rceil + 9$. □

A.2 Multi-Precision Multiplication

We instantiate a multi-precision multiplication circuit $C_{\times,i} \in \mathcal{C}_{\mathcal{R}_q,2w_p,2w_p}$ as described by Algorithms 12 and 13. The inputs are two vectors $\mathbf{a}, \mathbf{b} \in \mathcal{R}_q^{w_p}$

Algorithm 12 $\mathsf{SIMDMul}_{q,n,q_i,B_p}$

Parameter Settings: $q_i \mid q$, $(B_p-1)^2 < q_i$ and $w_p = \lceil \log_{B_p} q_i \rceil$.
Input: $\mathbf{a} := (a_1(X), \ldots, a_{w_p}(X))^T \in \mathcal{R}_q^{w_p}$, $\mathbf{b} := (b_1(X), \ldots, b_{w_p}(X))^T \in \mathcal{R}_q^{w_p}$.
Output: $\mathbf{c} \in \mathcal{R}_q^{2w_p}$

Initialize columns $\mathbf{C} := (\mathbf{c}_1, \ldots, \mathbf{c}_{2w_p})$ as empty lists
for $l \in [w_p]$ **do**
  for $r \in [w_p]$ **do**
    $t_{l,r}(X) \leftarrow a_l(X) \cdot b_r(X)$
    $\ell_{l,r}(X), h_{l,r}(X) \leftarrow \mathsf{Mod\&Floor}_{q,n,q_i,B_p}(t_{l,r}(X))$
    Append $\ell_{l,r}(X)$ to column $\mathbf{c}_{l+r-1}$
    Append $h_{l,r}(X)$ to column $\mathbf{c}_{l+r}$ ▷ for $l + r > 2w_p$ drop the high limb
  end for
end for
$(\mathbf{s}', \mathbf{c}') \leftarrow \mathsf{SIMDWallaceTreeCompress}_{q,n,q_i,B_p}(\mathbf{C})$
$\mathbf{u} \leftarrow \mathsf{SIMDAdd}_{q,n,q_i,B_p}(\mathbf{s}', \mathbf{c}'_{1..2w_p})$ ▷ Truncate $\mathbf{c}'$ before adding
$\mathbf{c} \leftarrow \mathbf{u}_{1..2w_p}$ ▷ Drop the top limb from the addition's result
return c

encoding, slotwise, the base-B_p limbs of two integers in $\mathbb{Z}_{q_i}^n$. The circuit outputs $\mathbf{c} \in \mathcal{R}_q^{2w_p}$ representing the $2w_p$-limb product of these big integers (per slot). Concretely, it forms all w_p^2 limbwise products $t_{l,j}(X) = a_l(X)b_j(X)$, splits each into low/high limbs via $\mathsf{Mod\&Floor}$, and stacks them into weight-aligned columns. The $\mathsf{SIMDWallaceTreeCompress}$ routine (Algorithm 13) compresses each column to at most two limbs using grouped additions followed by $\mathsf{Mod\&Floor}$, yielding two rows $(\mathbf{s}', \mathbf{c}')$. A final $\mathsf{SIMDAdd}$ adds these rows (with $\mathbf{c}'$ viewed as right-shifted by one limb) and the top carry limb is dropped to obtain exactly $2w_p$ limbs. By the ring-homomorphism of the evaluation slots, all computations proceed in SIMD fashion over $\mathbb{Z}_{q_i}^n$.

Algorithm 13 $\mathsf{SIMDWallaceTreeCompress}_{q,n,q_i,B_p}$

Parameter Settings: $q_i \mid q$, $(B_p-1)^2 < q_i$ and $w_p = \lceil \log_{B_p} q_i \rceil$.
Input: Columns $\mathbf{C} := (\mathbf{c}_1, \ldots, \mathbf{c}_t)$.
Output: $\mathbf{s}' \in \mathcal{R}_q^t$, $\mathbf{c}' \in \mathcal{R}_q^{t+1}$

while $\max_{k\in[t]} \text{height}(\mathbf{c}_k) > 2$ **do**
  $\mathbf{C}_{\text{next}} \leftarrow$ new vector of $t{+}1$ empty columns ▷ Extra tail for carries
  for $k \in [t]$ **do**
    while $\mathbf{C}_k$ is not empty **do**
      Pop up to $B_p{-}1$ limbs (or 3 when $B_p{=}2$) from $\mathbf{C}_k$ to form a group $\mathbf{H}$
      $u(X) \leftarrow \sum_{h\in\mathbf{H}} h$
      $u^{lo}(X), u^{hi}(X) \leftarrow \mathsf{Mod\&Floor}_{q,n,q_i,B_p}(u(X))$
      Append $u^{lo}(X)$ to column $\mathbf{c}_{\text{next},k}$
      Append $u^{hi}(X)$ to column $\mathbf{c}_{\text{next},k+1}$
    end while
  end for
  $\mathbf{C} \leftarrow (\mathbf{c}_{\text{next},1}, \ldots, \mathbf{c}_{\text{next},t})$ ▷ Discard tail column $\mathbf{C}_{\text{next},t+1}$
end while ▷ All columns now have height ≤ 2
Initialize $\mathbf{s}' \leftarrow (0, \ldots, 0)$ of size t
Initialize $\mathbf{c}' \leftarrow (0, \ldots, 0)$ of size $t{+}1$
for $k \in [t]$ **do**
  if $\text{height}(\mathbf{c}_k) = 1$ **then**
    $s'_k(X) \leftarrow c_{k,1}(X)$
  else if $\text{height}(\mathbf{c}_k) = 2$ **then**
    $v(X) \leftarrow c_{k,1}(X) + c_{k,2}(X)$
    $v^{lo}(X), v^{hi}(X) \leftarrow \mathsf{Mod\&Floor}_{q,n,q_i,B_p}(v(X))$
    $s'_k(X) \leftarrow v^{lo}(X)$
    $c'_{k+1}(X) \leftarrow v^{hi}(X)$
  else
    $s'_k(X) \leftarrow 0$
  end if
end for
return $(\mathbf{s}', \mathbf{c}')$ ▷ Carries are right-shifted by 1

Lemma 10. *We keep the same parameter settings in Definition 4. For every $i \in [h]$, there exists a circuit $C_{\times,i}$ implementing Algorithm 12 such that $C_{\times,i} \in \mathcal{C}_{\mathcal{R}_q,2w_p,2w_p}$ and the following properties hold:*

1. *The below set is (a subset of) the admissible input set of $C_{\times,i}$:*

$$\mathcal{X}_{C_{\times,i},B_{\times,C_{\times,i}}} := \left\{ \begin{pmatrix} \left(\mathsf{NTT}_i^{-1}(\tilde{\mathbf{a}}_l) \bmod q\mathcal{R} + q_i e_{a_l}(X)\right)_{l\in[w_p]}, \\ \left(\mathsf{NTT}_i^{-1}\left(\tilde{\mathbf{b}}_l\right) \bmod q\mathcal{R} + q_i e_{b_l}(X)\right)_{l\in[w_p]} \end{pmatrix} \right\},$$

   where $\tilde{\mathbf{a}}_l, \tilde{\mathbf{b}}_l \in [0, B_p-1]^n$ for all $l \in [w_p]$, the polynomials $e_{a_l}(X), e_{b_l}(X) \in \mathcal{R}_q$ satisfy $|e_{a_l}(X)|, |e_{b_l}(X)| \leq n-1$, and

$$B_{\times,C_{\times,i}} = (n(q_i-1))^{3^{\mathcal{O}(\lceil \log_2 w_p \rceil)}}.$$

2. *The non-free depth of* $C_{\times,i}$ *is bounded by*

$$2\lceil \log_c w_p \rceil + 2\lceil \log_2(2w_p) \rceil + 13,$$

where $c = s/2 > 1$ *is the compression rate with* $s = \max\{B_p - 1, 3\}$*, and its additive width is*

$$\max\{s-1, n\}.$$

3. *For any input* $\begin{pmatrix} (a_l(X))_{l\in[w_p]} \\ (b_l(X))_{l\in[w_p]} \end{pmatrix} \in \mathcal{X}_{C_{\times,i}, B_{\times,}, C_{\times,i}}$ *there exists* $\mathbf{e}_c \in \mathcal{R}_q^{2w_p}$ *such that*

$$\mathbf{c} = \left(\mathit{NTT}_i^{-1}\left([\tilde{\mathbf{c}}_l]_{q_i} \right) \bmod q\mathcal{R} \right)_{l\in[2w_p]}^T + q_i \mathbf{e}_c,$$
$$\|\mathbf{e}_c\|_\infty \le n-1,$$

where

$$\forall\, l \in [w_p]: \quad \tilde{\mathbf{a}}_l := \mathit{NTT}_i(a_l(X) \bmod q_i\mathcal{R}) \in \mathbb{Z}_{q_i}^n,$$
$$\forall\, l \in [w_p]: \quad \tilde{\mathbf{b}}_l := \mathit{NTT}_i(b_l(X) \bmod q_i\mathcal{R}) \in \mathbb{Z}_{q_i}^n,$$
$$\mathbf{c} := C_{\times,i}\begin{pmatrix} (a_l(X))_{l\in[w_p]} \\ (b_l(X))_{l\in[w_p]} \end{pmatrix}, \qquad \left([\tilde{\mathbf{c}}_l]_{q_i} \right)_{l\in[2w_p]}^T \in \mathbb{Z}_q^{n\times 2w_p},$$

and the evaluation-slot integers satisfy

$$\boxplus_{l=1}^{2w_p} B_p^{l-1}\, \tilde{\mathbf{c}}_l = \left(\boxplus_{l=1}^{w_p} B_p^{l-1}\, \tilde{\mathbf{a}}_l\right) \boxtimes \left(\boxplus_{l=1}^{w_p} B_p^{l-1}\, \tilde{\mathbf{b}}_l\right) \quad \textit{over integers.}$$

Proof. We prove the claim for the multiplication circuit $C_{x,i}$ *by decomposition into two stages.*

Stage 1: Partial products.

Functionality. *For every pair* $(l, r) \in [w_p] \times [w_p]$*, the circuit computes* $t_{l,r}(X) = a_l(X)\, b_r(X)$ *and executes* $\mathsf{Mod\&Floor}_{q,n,q_i,B_p}(t_{l,r}(X))$*. By Lemma 6, it follows that the outputs* $(\ell_{l,r}(X), h_{l,r}(X))$ *correspond to the low/high limbs of* $t_{l,r}(X)$ *in the evaluation slots, respectively, as follows:*

$$\ell_{l,r}(X) = \mathit{NTT}_i^{-1}\left(\left(\tilde{t}_{l,r,j} \bmod B_p \right)_{j\in[n]} \right) \bmod q\mathcal{R} + q_i\, e_{\ell_{l,r}}(X),$$
$$h_{l,r}(X) = \mathit{NTT}_i^{-1}\left(\left(\left\lfloor \frac{\tilde{t}_{l,r,j}}{B_p} \right\rfloor \right)_{j\in[n]} \right) \bmod q\mathcal{R} + q_i\, e_{h_{l,r}}(X).$$

where $(\tilde{t}_{l,r,j})_{j\in[n]} = \mathit{NTT}_i([t_{l,r}(X)]_{q_i})$ *and* $|e_{\ell_{l,r}}(X)|, |e_{h_{l,r}}(X)| \le n-1$*.* $\ell_{l,r}(X)$ *is appended to column* $\mathbf{c}_{l+r-1}$ *and* $h_{l,r}(X)$ *to column* $\mathbf{c}_{l+r}$ *(weight alignment). After Stage 1, the multiset of columns* $\mathbf{C} = (\mathbf{c}_1, \ldots, \mathbf{c}_{2w_p})$ *encodes the (unreduced) partial products with correct weights in every evaluation slot.*

Input admissibility. *By the admissible-input hypothesis,* $[\tilde{\mathbf{a}}_l]_{q_i}, [\tilde{\mathbf{b}}_r]_{q_i} \in [0, B_p - 1]^n$. *Hence for every* (l, r) *and every* $j \in [n]$,

$$0 \le \tilde{t}_{l,r,j} = \tilde{a}_{l,j}\, \tilde{b}_{r,j} \le (B_p - 1)^2 < q_i,$$

so $\left\| \left[\tilde{\mathbf{a}}_l \boxtimes \tilde{\mathbf{b}}_r\right]_{q_i} \right\|_\infty \le (B_p{-}1)^2$. *Therefore each of the* w_p^2 *calls to* $\mathsf{Mod\&Floor}_{q,n,q_i,B_p}(t_{l,r}(X))$ *is within the admissible domain of* $C_{\mathsf{Mod\&Floor},i}$ *(Lemma 6) and never returns* $\perp$.

Bound of the left-hand inputs to multiplication gates. *The only polynomial-to-polynomial multiplications in this stage are the* $t_{l,r}(X) = a_l(X)b_r(X)$. *We fix the convention that the* left-hand *operand is* $a_l(X)$ *for all such gates. From the admissible-input form* $a_l(X) = \mathsf{NTT}_i^{-1}(\tilde{\mathbf{a}}_l) \bmod q\mathcal{R} + q_i e_{a_l}(X)$ *with* $|\mathsf{NTT}_i^{-1}(\tilde{\mathbf{a}}_l) \bmod q\mathcal{R}| \le q_i - 1$ *and* $|e_{a_l}(X)| \le n - 1$, *it follows that*

$$|a_l(X)| \le \left|\mathsf{NTT}_i^{-1}(\tilde{\mathbf{a}}_l) \bmod q\mathcal{R}\right| + q_i\, |e_{a_l}(X)| \le (q_i - 1) + (n - 1)q_i \le nq_i - 1.$$

Hence the bound on all left-hand inputs to multiplication gates in Stage 1 can be taken as

$$B_{\times, C_{\times,i},1} = nq_i - 1.$$

(Any constant multiplications internal to $C_{\mathsf{Mod\&Floor},i}$ *do not contribute to this bound; see Lemma 6.)*

Non-free depth and additive width. *For each pair* (l, r) *the computation path consists of one polynomial multiplication* $a_l(X)b_r(X)$ *(one non-free layer), followed by a single call to* $C_{\mathsf{Mod\&Floor},i}$ *(non-free depth* 2 *by Lemma 6). All* w_p^2 *pairs are processed in parallel, so the Stage 1 non-free depth is* 3. *The additive width is inherited from* $C_{\mathsf{Mod\&Floor},i}$ *and equals* n.

Stage 2: Wallace-tree compression [18] and final addition.

Functionality. *Let* $t := 2w_p$ *denote the number of columns. After Stage 1 we have a multiset of columns* $\mathbf{C} = (\mathbf{c}_1, \ldots, \mathbf{c}_t)$ *where each column* $\mathbf{c}_k$ *contains polynomials encoding slotwise digits in* $[0, B_p{-}1]$ *according to the output of the previous stage. One compression* round *of Algorithm 13 processes each column* k *by repeatedly forming a group* $\mathbf{H} \subseteq \mathbf{c}_k$ *of size at most* $B_p{-}1$ *(or 3 when* $B_p{=}2$*), summing* $u(X) := \sum_{h \in \mathbf{H}} h$, *and splitting* u *via one call to* $\mathsf{Mod\&Floor}_{q,n,q_i,B_p}$ *as*

$$(u^{lo}(X), u^{hi}(X)) \leftarrow \mathsf{Mod\&Floor}_{q,n,q_i,B_p}(u(X)).$$

The algorithm appends $u^{lo}(X)$ *to column* k *of the* next *layer and* $u^{hi}(X)$ *to column* $k{+}1$ *of the next layer. Discarding the tail column* $t{+}1$ *amounts to keeping only the least significant* $t = 2w_p$ *limbs. Rounds repeat until every column has height* ≤ 2.

In the final pass of Algorithm 13, each column k *with height* 2 *is reduced by computing* $v(X) := c_{k,1}(X){+}c_{k,2}(X)$ *and* $(v^{lo}(X), v^{hi}(X)) \leftarrow \mathsf{Mod\&Floor}_{q,n,q_i,B_p}(v(X))$, *then setting* $s'_k(X) := v^{lo}(X)$ *and* $c'_{k+1}(X) := v^{hi}(X)$; *when the height is* 1 *we set* $s'_k(X)$ *to that single limb and* $c'_{k+1}(X) := 0$. *Finally, Algorithm 12 invokes*

$\mathsf{SIMDAdd}_{q,n,q_i,B_p}$ *on* $(\mathbf{s}', \mathbf{c}'_{1..t})$ *and drops the top limb, yielding exactly the* $2w_p$ *least significant base-*B_p *limbs of the product in every evaluation slot, as claimed in Item (3).*

Bound of the left-hand inputs to multiplication gates. *Within the Wallace-tree compression, all non-free work is performed by calls to* $C_{\mathsf{Mod\&Floor},i}$*, which (by Lemma 6) contribute no polynomial-to-polynomial multiplication on their left-hand inputs, hence they do not increase the bound. The only polynomial-to-polynomial multiplications in Stage 1I occur inside the* KoggeStonePrefixSum *subroutine of* SIMDAdd *applied to length* $t = 2w_p$*. It follows that the bound on all left-hand inputs to multiplication gates in Stage 1I is*

$$B_{\times, C_{\times,i},2} = \big(n(q_i{-}1)\big)^{3^{\mathcal{O}(\lceil \log_2(2w_p)\rceil)}} = \big(n(q_i{-}1)\big)^{3^{\mathcal{O}(\lceil \log_2 w_p\rceil)}}.$$

Non-free depth and additive width. *Each compression* round *applies a single* Mod&Floor *per group, hence contributes non-free depth* 2*. Let* $s := \begin{cases} B_p - 1, & B_p \geq 3, \\ 3, & B_p = 2, \end{cases}$ *be the per-column group size used in Algorithm 13. Every group of size* s *produces two outputs (*u^{lo} *in column* k *and* u^{hi} *in column* $k{+}1$*), so the maximal column height shrinks by at least a factor*

$$c := \frac{s}{2} > 1 \qquad \textit{(“compression rate”).}$$

Therefore the number of rounds needed to reduce the maximum height from $\mathcal{O}(w_p)$ *down to* ≤ 2 *is at most* $\lceil \log_c w_p \rceil$*. The final pass that reduces height-*2 *columns adds another depth-*2*. The concluding call* $\mathsf{SIMDAdd}_{q,n,q_i,B_p}(\mathbf{s}', \mathbf{c}'_{1..2w_p})$ *has non-free depth* $2\lceil \log_2(2w_p)\rceil + 8$ *by Lemma 9 (with* $w_p \mapsto 2w_p$*). Therefore Stage 1I contributes non-free depth*

$$2\big\lceil \log_c w_p \big\rceil + 2 + \big(2\lceil \log_2(2w_p)\rceil + 8\big)$$

.

For the additive width, in each round and for each column, we form $u(X) = \sum_{h\in\mathbf{H}} h$ *with* $|\mathbf{H}| \leq s$*, which gives an additive width of at most* $s{-}1$ *additions. The final height-*2 *reduction performs one addition per active column (width* $\leq n$*), and the concluding* SIMDAdd *also has width* n*. We conclude that Stage 1I has additive width* $\max\{s{-}1, 1, n\} = \max\{s{-}1, n\}$*.*

Input admissibility. *All Stage 1I inputs are* Mod&Floor *outputs from Stage 1, hence encode digits in* $[0, B_p{-}1]$ *with* q_i*-scaled error* $\leq n{-}1$ *(Lemma 6). In each compression round, a group* $\mathbf{H}$ *has size* $|\mathbf{H}| \leq s$*, so slotwise* $\tilde{u}_j = \sum_{h\in\mathbf{H}} \tilde{h}_j \leq s(B_p{-}1)$*. For* $B_p \geq 3$ *we have* $s \leq B_p{-}1$*, thus* $\tilde{u}_j \leq (B_p{-}1)^2$*, which lies in the admissible domain of* $C_{\mathsf{Mod\&Floor},i}$*. In the final pass on height-2 columns,* $\tilde{v}_j \leq 2(B_p{-}1)$*, hence also admissible. The inputs to the concluding* $\mathsf{SIMDAdd}_{q,n,q_i,B_p}(\mathbf{s}', \mathbf{c}'_{1..2w_p})$ *are again* Mod&Floor *outputs, so the admissibility conditions required by Lemma 9 (with length* $t = 2w_p$*) are met; therefore no LUT returns* $\perp$*.*

Algorithm 14 $\mathsf{SIMDAddMod}_{q,n,q_i,B_p}$

Parameter Settings: $q_i \mid q$, $(B_p-1)^2 < q_i$ and $w_p = \lceil \log_{B_p} q_i \rceil$.
Input: $\mathbf{a} := (a_1(X), \ldots, a_{w_p}(X)) \in \mathcal{R}_q^{w_p}$, $\mathbf{b} := (b_1(X), \ldots, b_{w_p}(X)) \in \mathcal{R}_q^{w_p}$, Modulus $\mathbf{m} := (m_1(X), \ldots, m_{w_p}(X)) \in \mathcal{R}_q^{w_p}$.
Output: Result $\mathbf{s} := (s_1(X), \ldots, s_{w_p}(X)) \in \mathcal{R}_q^{w_p}$.

1: $\mathbf{s}' \leftarrow \mathsf{SIMDAdd}_{q,n,q_i,B_p}(\mathbf{a}, \mathbf{b})$ ▷ $\mathbf{s}'$ has w_p+1 limbs
2: $\mathbf{m}_{pad} \leftarrow (\mathbf{m}, 0)$ ▷ Pad modulus to w_p+1 limbs
3: $(\mathbf{d}, c(X)) \leftarrow \mathsf{SIMDSub}_{q,n,q_i,B_p}(\mathbf{s}', \mathbf{m}_{pad})$
4: $sel(X) \leftarrow 1 - c(X)$
5: **for** $l \in [w_p]$ **do**
6: $\quad s_l(X) \leftarrow sel(X) \cdot d_l(X) + (1 - sel(X)) \cdot s'_l(X)$
7: **end for**
8: **return s**

Properties of $C_{x,i}$.
We prove that the three properties of $C_{x,i}$ hold. For Item (1), by the input admissibility in the first and second stages, every call to Mod&Floor and to SIMDAdd lies within the admissible domain, hence no LUT returns $\perp$. The bound on left-hand inputs to multiplication gates is $B_{\times,C_{x,i}} = \max\{B_{\times,C_{x,i},1}, B_{\times,C_{x,i},2}\} = (n(q_i-1))^{3^{\mathcal{O}(\lceil \log_2 w_p \rceil)}}$. For Item (2), summing the non-free depths of the two stages gives

$$\mathrm{depth}_{\mathrm{nf}}(C_{x,i}) \le \underbrace{3}_{\textit{Stage I}} + \underbrace{\Big(2\lceil \log_c w_p \rceil + 2 + 2\lceil \log_2(2w_p) \rceil + 8\Big)}_{\textit{Stage II}} = 2\lceil \log_c w_p \rceil + 2\lceil \log_2(2w_p) \rceil + 13,$$

where $c = s/2 > 1$ is the per-round compression rate with $s = \max\{B_p - 1, 3\}$. The additive width is $\max\{s-1, n\}$, accounting for the within-group sums $\sum_{h \in \mathbf{H}} h$ (costing at most $s-1$ parallel additions per column) and the width n inherited from Mod&Floor and SIMDAdd. Item (3) follows from the correctness of Stages 1 and 2, as argued above.

A.3 Modular Addition and Subtraction

Lemma 11. *We keep the same parameter settings in Definition 4. For each $i \in [h]$, there exist three circuits, denoted by $C_{+,q_i} \in \mathcal{C}_{\mathcal{R}_q,2w_p,w_p}$, $C_{-,q_i} \in \mathcal{C}_{\mathcal{R}_q,2w_p,w_p}$, implementing, respecitvely, Algorithm 14 and Algorithm 15 such that the following properties hold:*

1. *The below set is (a subset of) the admissible input sets of $C_{\pm,q_i}$ in common:*

$$\mathcal{X}_{C_{q_i},B_\times,C_{\pm,q_i}} := \left\{ \begin{pmatrix} \left(NTT_i^{-1}(\tilde{\mathbf{a}}_l) \mod q\mathcal{R} + q_i e_{a_{i,l}}(X)\right)_{l\in[w_p]}, \\ \left(NTT_i^{-1}\left(\tilde{\mathbf{b}}_l\right) \mod q\mathcal{R} + q_i e_{b_{i,l}}(X)\right)_{l\in[w_p]} \end{pmatrix} \right\}_{\substack{l\in[w_p],\mathbf{a}_l\in[0,B_p-1]^n,\\ \mathbf{b}_l\in[0,B_p-1]^n,|e_{a_{i,l}}(X)|\le n-1,\\ |e_{b_{i,l}}(X)|\le n-1}},$$

Algorithm 15 $\mathsf{SIMDSubMod}_{q,n,q_i,B_p}$

Parameter Settings: $q_i \mid q$, $(B_p-1)^2 < q_i$ and $w_p = \lceil \log_{B_p} q_i \rceil$.

Input: $\mathbf{a} := (a_1(X), \ldots, a_{w_p}(X)) \in \mathcal{R}_q^{w_p}$, $\mathbf{b} := (b_1(X), \ldots, b_{w_p}(X)) \in \mathcal{R}_q^{w_p}$, Modulus $\mathbf{m} := (m_1(X), \ldots, m_{w_p}(X)) \in \mathcal{R}_q^{w_p}$.

Output: Result $\mathbf{s} := (s_1(X), \ldots, s_{w_p}(X)) \in \mathcal{R}_q^{w_p}$.

1: $(\mathbf{d}, c(X)) \leftarrow \mathsf{SIMDSub}_{q,n,q_i,B_p}(\mathbf{a}, \mathbf{b})$
2: $\mathbf{s}' \leftarrow \mathsf{SIMDAdd}_{q,n,q_i,B_p}(\mathbf{d}, \mathbf{m})$ ▷ $\mathbf{s}'$ has $w_p + 1$ limbs
3: **for** $l \in [w_p]$ **do**
4: $\quad s_l(X) \leftarrow c(X) \cdot s'_l(X) + (1 - c(X)) \cdot d_l(X)$
5: **end for**
6: **return s**

where $B_{\times, C_{\pm,q_i}} = B_{\times, C_{\pm,i}} = (n(q_i-1))^{3^{\mathcal{O}(\lceil \log_2 w_p \rceil)}}$.

2. *The non-free depth of* $C_{\pm,q_i}$ *is bounded by* $4\lceil \log_2 w_p \rceil + 18$. *The additive width of both circuits is* n.
3. *For any vector* $\begin{pmatrix} (a_l(X))_{l\in[w_p]} \\ (b_l(X))_{l\in[w_p]} \end{pmatrix} \in \mathcal{X}_{C_{\pm,i}, B_{\times,q_i}}$, *there exist* $\mathbf{e}_c \in \mathcal{R}_q^{w_p}$ *such that*

$$\mathbf{c} = \left(\mathsf{NTT}_i^{-1}\left([\tilde{\mathbf{c}}_l]_{q_i} \right) \mod q\mathcal{R} \right)_{l\in[w_p]}^T + q_i \mathbf{e}_c,$$
$$\|\mathbf{e}_c\|_\infty \leq n-1$$

hold, where

$$\forall l \in [w_p] \quad \tilde{\mathbf{a}}_l := \mathsf{NTT}_i(a_l(X) \mod q_i\mathcal{R}) \in \mathbb{Z}_{q_i}^n,$$
$$\forall l \in [w_p] \quad \tilde{\mathbf{b}}_l := \mathsf{NTT}_i(b_l(X) \mod q_i\mathcal{R}) \in \mathbb{Z}_{q_i}^n,$$

$$\mathbf{c} := C_{+,q_i} \begin{pmatrix} (a_l(X))_{l\in[w_p]} \\ (b_l(X))_{l\in[w_p]} \end{pmatrix},$$
$$\left([\tilde{\mathbf{c}}_l]_{q_i} \right)_{l\in[w_p]}^T \in \mathbb{Z}_q^{n\times(w_p)}$$
$$s.t.\ \boxplus_{l\in[w_p+1]} B_p^{l-1}\tilde{\mathbf{c}}_l = \left(\boxplus_{l\in[w_p]} B_p^{l-1}\tilde{\mathbf{a}}_l\right) \boxplus \left(\boxplus_{l\in[w_p]} B_p^{l-1}\tilde{\mathbf{b}}_l\right) \mod q_i.$$

4. *For any vector* $\begin{pmatrix} (a_l(X))_{l\in[w_p]} \\ (b_l(X))_{l\in[w_p]} \end{pmatrix} \in \mathcal{X}_{C_{\pm,i}, B_{\times,q_i}}$, *there exist* $\mathbf{e}_c \in \mathcal{R}_q^{w_p}$ *such that*

$$\mathbf{c} = \left(\mathsf{NTT}_i^{-1}\left([\tilde{\mathbf{c}}_l]_{q_i} \right) \mod q\mathcal{R} \right)_{l\in[w_p]}^T + q_i \mathbf{e}_c,$$
$$\|\mathbf{e}_c\|_\infty \leq n-1$$

hold, where

$$\forall l \in [w_p] \quad \tilde{\mathbf{a}}_l := \mathsf{NTT}_i(a_l(X) \mod q_i\mathcal{R}) \in \mathbb{Z}_{q_i}^n,$$
$$\forall l \in [w_p] \quad \tilde{\mathbf{b}}_l := \mathsf{NTT}_i(b_l(X) \mod q_i\mathcal{R}) \in \mathbb{Z}_{q_i}^n,$$

$$\mathbf{c} := C_{-,q_i}\begin{pmatrix}(a_l(X))_{l\in[w_p]}\\(b_l(X))_{l\in[w_p]}\end{pmatrix},$$
$$\left([\tilde{\mathbf{c}}_l]_{q_i}\right)^T_{l\in[w_p]} \in \mathbb{Z}_q^{n\times(w_p)}$$
$$s.t.\ \boxplus_{l\in[w_p+1]} B_p^{l-1}\tilde{\mathbf{c}}_l = \left(\boxplus_{l\in[w_p]} B_p^{l-1}\tilde{\mathbf{a}}_l\right) \boxminus \left(\boxplus_{l\in[w_p]} B_p^{l-1}\tilde{\mathbf{b}}_l\right) \mod q_i.$$

Proof. We sketch the case of C_{+,q_i}; the proof for C_{-,q_i} is analogous.

Functionality: *The modular addition circuit C_{+,q_i}, performs algorithm 14 SIMDAddMod: given two w_p-limb base-B_p inputs $\mathbf{a} = (a_1, \ldots, a_{w_p})$ and $\mathbf{b} = (b_1, \ldots, b_{w_p})$ (each limb in R_q encodes n SIMD slots modulo q_i), it first computes the unreduced sum $\mathbf{s}' \leftarrow \mathsf{SIMDAdd}(\mathbf{a}, \mathbf{b})$, which returns $w_p + 1$ limbs. It then subtracts the (padded) modulus $\mathbf{m}_{\text{pad}}$ via $(\mathbf{d}, c(X)) \leftarrow \mathsf{SIMDSub}(\mathbf{s}', \mathbf{m}_{\text{pad}})$, obtaining a borrow bit-polynomial $c(X) \in R_q$ that encodes, slotwise, whether $\mathsf{s}' \geq \mathsf{m}$ (no borrow) or $\mathsf{s}' < \mathsf{m}$ (borrow). Finally it forms $\mathsf{sel} := 1 - c(X)$ and outputs, for each limb $l \in [w_p]$,*

$$s_l := \mathsf{sel} \cdot d_l + (1 - \mathsf{sel}) \cdot s'_l,$$

i.e., it selects d when $s' \geq m$ and s' otherwise. The subroutines SIMDAdd and SIMDSub are given in Appendix A, and their admissibility, correctness and depth bounds are summarized in Lemma 9.

Input admissibility: *Each limb of the inputs to C_{+,q_i} has the admissible form*

$$a_l = \mathsf{NTT}_i^{-1}\left([\mathbf{a}_l]_{q_i}\right) \bmod q\mathcal{R} + q_i\, e_{a,l}, \qquad b_l = \mathsf{NTT}_i^{-1}\left([\mathbf{b}_l]_{q_i}\right) \bmod q\mathcal{R} + q_i\, e_{b,l},$$

with $[\mathbf{a}_l]_{q_i}, [\mathbf{b}_l]_{q_i} \in [0, B_p - 1]^n$ and $\|e_{a,l}\|_\infty, \|e_{b,l}\|_\infty \leq n-1$. By the admissibility guarantees assumed in Lemma 9 for the subroutines SIMDAdd and SIMDSub (Appendix A), the above inputs are admissible for every call that C_{+,q_i} makes to these subroutines, so no LUT evaluation ever returns $\perp$.

Bound of the left-hand inputs to multiplication gates: *The multiplication bound for the subroutines SIMDAdd and SIMDSub is equal to $B_{\times,C_{\pm,i}} \geq nq_i$ (Lemma 9). The final multiplication layer in C_{+,q_i} computes, for each limb l,*

$$s_l = \mathsf{sel} \cdot d_l + (1 - \mathsf{sel}) \cdot s'_l \quad with \quad \mathsf{sel} = 1 - c(X),$$

where $d_l(X)$ and $s'_l(X)$ are the outputs of Mod&Floor and therefore bounded by nq_i. Hence the final multiplication layer does not increase the bound.

Non-free depth and additive width: *The non-free depth of C_{+,q_i} is the sum of the depths of its stages:*

$$\text{depth}\left(C_{+,q_i}\right) = \text{depth}\left(\mathsf{SIMDAdd}\right) + \text{depth}\left(\mathsf{SIMDSub}\right) + 1,$$

where the final "+1" accounts for the single multiplicative layer used by the selector $s_l(X) = (1 - c(X))\, d_l(X) + c(X)\, s'_l(X)$ (the trailing addition is free). The specific depths of SIMDAdd and SIMDSub are those certified by Lemma 9. The circuit also inherits the maximum additive width of its subroutines, which is n.

Properties of $C_{+,i}$.
Inputs satisfy Lemma 9, so all calls to SIMDAdd and SIMDSub are admissible; no LUT returns $\perp$. The multiplication bound is $B_{\times,C_{+,q_i}} = (n(q_i-1))^{3^{\mathcal{O}(\lceil \log_2 w_p \rceil)}}$, inherited from the subroutines. The non-free depth is $2\lceil \log_2 w_p \rceil + 8$, the sum of the depths of SIMDAdd and SIMDSub (Lemma 9) plus one layer for the final selector. The additive width is n.The third property follows from the correctness of SIMDAdd and SIMDSub (Lemma 9) and the definition of the selector.

A.4 Modular Multiplication

We adopt Montgomery modular multiplication [16, §14.3.2]. to avoid explicit division by the modulus. Operands are maintained in Montgomery form $x \mapsto xR \bmod m$ (with $R = B_p^{w_p}$), a schoolbook product is computed, and a REDC-style reduction with the precomputed constant $m' \equiv -m^{-1} \bmod B_p$ yields $xyR^{-1} \bmod m$ slotwise.

Algorithm 16 $\mathsf{MulMod}_{q,n,q_i,B_p}$

Parameter Settings: $q_i \mid q$, $(B_p - 1)^2 < q_i$ and $w_p = \lceil \log_{B_p} q_i \rceil$.

Input: Montgomery-form inputs $\mathbf{a}_m, \mathbf{b}_m \in \mathcal{R}_q^{w_p}$, Modulus $\mathbf{m} \in \mathcal{R}_q^{w_p}$, Montgomery constant $\mathbf{m}' \in \mathcal{R}_q^{w_p}$.

Output: Result $\mathbf{c}_m \in \mathcal{R}_q^{w_p}$.

1: $\mathbf{t} \leftarrow \mathsf{SIMDMul}_{q,n,q_i,B_p}(\mathbf{a}_m, \mathbf{b}_m)$ ▷ $\mathbf{t}$ has $2w_p$ limbs
2: $\mathbf{c}_m \leftarrow \mathsf{SIMDMontgomeryReduce}_{q,n,q_i,B_p}(\mathbf{t}, \mathbf{m}, \mathbf{m}')$
3: **return** $\mathbf{c}_m$

Algorithm 17 $\mathsf{SIMDMontgomeryReduce}_{q,n,q_i,B_p}$

Parameter Settings: $q_i \mid q$, $(B_p-1)^2 < q_i$ and $w_p = \lceil \log_{B_p} q_i \rceil$.

Input: $\mathbf{t} \in \mathcal{R}_q^{2w_p}$, Modulus $\mathbf{m} \in \mathcal{R}_q^{w_p}$, Montgomery constant $\mathbf{m}' \in \mathcal{R}_q^{w_p}$.

Output: Result $\mathbf{a} \in \mathcal{R}_q^{w_p}$.

$\mathbf{t}_{low} \leftarrow (t_1(X), \ldots, t_{w_p}(X))$ ▷ Lower w_p limbs of $\mathbf{t}$

$\mathbf{u}_{full} \leftarrow \mathsf{SIMDMul}_{q,n,q_i,B_p}(\mathbf{t}_{low}, \mathbf{m}')$

$\mathbf{u} \leftarrow (u_{full,1}(X), \ldots, u_{full,w_p}(X))$ ▷ Lower w_p limbs of the product

$\mathbf{p} \leftarrow \mathsf{SIMDMul}_{q,n,q_i,B_p}(\mathbf{u}, \mathbf{m})$

$\mathbf{v} \leftarrow \mathsf{SIMDAdd}_{q,n,q_i,B_p}(\mathbf{t}, \mathbf{p})$ ▷ $\mathbf{v}$ has $2w_p+1$ limbs

$\mathbf{a}' \leftarrow (v_{w_p+1}(X), \ldots, v_{2w_p+1}(X))$ ▷ Upper w_p+1 limbs of $\mathbf{v}$

$\mathbf{m}_{pad} \leftarrow (\mathbf{m}, 0)$ ▷ Pad modulus to w_p+1 limbs

$(\mathbf{d}, b(X)) \leftarrow \mathsf{SIMDSub}_{q,n,q_i,B_p}(\mathbf{a}', \mathbf{m}_{pad})$ ▷ $b(X)=0 \iff \mathbf{a}' \geq \mathbf{m}_{pad}$

$sel(X) \leftarrow 1 - b(X)$

for $i \in [w_p]$ **do**

  $a_i(X) \leftarrow sel(X) \cdot d_i(X) + (1 - sel(X)) \cdot a'_i(X)$

end for

return a

Lemma 12. *We keep the same parameter settings in Definition 4. Fix $i \in [h]$ and let the (per-slot) modulus $\mathbf{m} \in R_q^{w_p}$ and Montgomery constant $\mathbf{m}' \in R_q^{w_p}$ be circuit constants encoded limbwise with digits in $[0, B_p-1]$. and 17 such that $C_{\times,q_i} \in \mathcal{C}_{\mathcal{R}_q, 2w_p, w_p}$ and:*

1. *A subset of the admissible input set of $C_{\times,q_i}$ is*

$$\mathcal{X}_{C_{\times,q_i},B_{\times,C_{\times,q_i}}} := \left\{ \begin{pmatrix} \big(\mathit{NTT}_i^{-1}(\tilde{\mathbf{a}}_\ell) \bmod q\mathcal{R} + q_i\, e_{a_\ell}(X)\big)_{\ell\in[w_p]}, \\ \big(\mathit{NTT}_i^{-1}(\tilde{\mathbf{b}}_\ell) \bmod q\mathcal{R} + q_i\, e_{b_\ell}(X)\big)_{\ell\in[w_p]} \end{pmatrix} \right\},$$

   where $\tilde{\mathbf{a}}_\ell, \tilde{\mathbf{b}}_\ell \in [0, B_p-1]^n$ for all ℓ, and $e_{a_\ell}(X), e_{b_\ell}(X) \in R_q$ satisfy $|e_{a_\ell}(X)|, |e_{b_\ell}(X)| \leq n-1$. Moreover,

$$B_{\times,C_{\times,q_i}} = B_{\times,C_{\times,i}} = \big(n(q_i-1)\big)^{3^{\mathcal{O}(\lceil \log_2 w_p \rceil)}}.$$

2. *The non-free depth of $C_{\times,q_i}$ is bounded by*

$$\underbrace{3(2\lceil \log_c w_p \rceil + 2\lceil \log_2(2w_p) \rceil + 13)}_{\text{three calls to } \mathsf{SIMDMul}} + \underbrace{\big(2\lceil \log_2(2w_p) \rceil + 8\big)}_{\mathsf{SIMDAdd}} + \underbrace{\big(2\lceil \log_2(w_p+1) \rceil + 9\big)}_{\mathsf{SIMDSub}} + \underbrace{1}_{\text{final selector}},$$

   where $c = s/2 > 1$ with $s = \max\{B_p-1, 3\}$. Its additive width is $\max\{s-1, n\}$.

3. *For any input $\big((a_\ell(X))_{\ell\in[w_p]}, (b_\ell(X))_{\ell\in[w_p]}\big) \in \mathcal{X}_{C_{\times,q_i},B_{\times,C_{\times,q_i}}}$ there exists $\mathbf{e}_c \in R_q^{w_p}$ with $\|\mathbf{e}_c\|_\infty \leq n-1$ such that*

$$\mathbf{c} = \Big(\mathit{NTT}_i^{-1}\big([\tilde{\mathbf{c}}_\ell]_{q_i}\big) \bmod q\mathcal{R}\Big)_{\ell\in[w_p]}^T + q_i\, \mathbf{e}_c,$$

where, writing

$$\tilde{\mathbf{a}}_\ell := \mathsf{NTT}_i(a_\ell(X) \bmod q_i\mathcal{R}), \quad \tilde{\mathbf{b}}_\ell := \mathsf{NTT}_i(b_\ell(X) \bmod q_i\mathcal{R}),$$

and letting

$$A := \boxplus_{\ell=1}^{w_p} B_p^{\ell-1}\tilde{\mathbf{a}}_\ell, \quad B := \boxplus_{\ell=1}^{w_p} B_p^{\ell-1}\tilde{\mathbf{b}}_\ell, \quad M := \boxplus_{\ell=1}^{w_p} B_p^{\ell-1}\tilde{\mathbf{m}}_\ell, \quad R := B_p^{w_p},$$

the slotwise integers encoded by $\left(\tilde{\mathbf{c}}_\ell\right)_{\ell\in[w_p]}$ *satisfy*

$$\boxplus_{\ell=1}^{w_p} B_p^{\ell-1}\,\tilde{\mathbf{c}}_\ell \equiv A\,B\,R^{-1} \pmod{M}, \qquad 0 \le \boxplus_{\ell=1}^{w_p} B_p^{\ell-1}\,\tilde{\mathbf{c}}_\ell < M,$$

i.e., $C_{\times,q_i}$ returns the Montgomery product of the inputs (which remain in Montgomery form).

Proof. Admissibility and multiplication bound. *The inputs match the admissible form used in Lemma 10 and Lemma 9. Each call to* SIMDMul *(for* $\mathbf{t}$*,* $\mathbf{u}_{full}$*, and* $\mathbf{p}$*) takes limb-vectors whose digits lie in* $[0, B_p{-}1]$ *(either from constants* $\mathbf{m}, \mathbf{m}'$ *or from previous* Mod&Floor *outputs), hence all* Mod&Floor *subcalls are admissible (Lemma 6) and no LUT returns* $\perp$*. The same holds for the subsequent* SIMDAdd *and* SIMDSub *(Lemma 9). Therefore the bound on all left-hand inputs to multiplication gates is inherited from Lemma 10, namely* $B_{\times,C_{\times,q_i}} = B_{\times,C_{\times,i}} = (n(q_i{-}1))^{3^{\mathcal{O}(\lceil\log_2 w_p\rceil)}}$.

Depth and width. *Summing the certified depths of the subroutines yields the stated bound: three calls to* SIMDMul *contribute* $3(2\lceil\log_c w_p\rceil + 2\lceil\log_2(2w_p)\rceil + 13)$ *(Lemma 10), the single* SIMDAdd *on length* $2w_p$ *contributes* $2\lceil\log_2(2w_p)\rceil + 8$*, the* SIMDSub *on length* $w_p{+}1$ *contributes* $2\lceil\log_2(w_p{+}1)\rceil{+}9$ *(Lemma 9), and the final selector adds one multiplicative layer. The additive width is* $\max\{s{-}1, n\}$*, inherited from Lemma 10 (for the grouped sums) and Lemma 9.*

Correctness and error bound. *By Lemma 10, each* SIMDMul *returns limbs whose evaluation slots encode the intended integer products, with* q_i*-scaled error polynomials bounded by* $n{-}1$ *(Lemma 6);* SIMDAdd/SIMDSub *preserve this form and error bound (Lemma 9). The Montgomery reduction step satisfies the standard identity* $\mathrm{REDC}(t) \equiv t\,R^{-1} \pmod{M}$ *per slot. Since inputs are in Montgomery form,* $t = (AR)(BR) = ABR^2$*, hence the returned value encodes* $ABR \bmod M$*; the conditional subtraction (using the borrow polynomial from* SIMDSub*) ensures the result lies in* $[0, M)$*. Therefore there exists* $\mathbf{e}_c \in R_q^{w_p}$ *with* $\|\mathbf{e}_c\|_\infty \le n{-}1$ *such that the output limbs represent the per-slot Montgomery product, as stated.*